DOCUMENT RESUME

ED 022 175

AL 001 473

By-Lindsay, Robert K.

A HEURISTIC PARSING PROCEDURE FOR A LANGUAGE LEARNING PROGRAM.

Texas Univ., Austin.

Report No-IPR-12

Pub Date 28 May 64

Note-93p.

EDRS Price MF-$0.50 HC-$3.80

Descriptors-*COMPUTATIONAL LINGUISTICS, *COMPUTER PROGRAMS, CONTEXT FREE GRAMMAR, DATA ANALYSIS, PHRASE STRUCTURE, SEMANTICS, *SYNTAX

Identifiers-*Labeled Dependency Trees

This paper reports a portion of a research effort to develop a program which will simulate the language learning behavior of humans. Here presented is a heuristic parsing procedure which accepts natural language sentences and produces for each a form of analysis called a "labeled dependency tree." The formal grammar on which the procedure is based differs from the "phrase structure" formalism of Chomsky (1957), and the analysis procedure attempts to discover the single most probable analysis rather than all analyses of ambiguous sentences. Included are discussions of the syntax-meaning distinction, the special problems of simulation, the need to handle a general class of inputs, and the need for analysis procedures which are to be self-organizing. The paper also describes a computer program for analysis of sentences and reports an experiment with the program. (Author/DO)

INFORMATION PROCESSING REPORT NUMBER 12

A HEURISTIC PARSING PROCEDURE FOR A LANGUAGE LEARNING PROGRAM

Robert K. Lindsay

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

ED022175

[redacted]

THE UNIVERSITY OF TEXAS
Austin, Texas
May 28, 1964

AL 001 473

TABLE OF CONTENTS

Abstract ii

Preface iii

Introduction 1

Chomsky's formalism and the problems it raises 2

The distinction between syntax and meaning 8

Requirements for a language processor 11

JIGSAW 13

    The interpretation 17

    The dictionary 20

    The syntax directory and syntax routines 21

    The teacher 23

    Cognitive limits 24

JIGSAW-1 24

    Labelled dependency trees 25

    The dictionary 27

    The syntax directory 29

    Cognitive limits 31

    Initializing the program 31

    Program outline 32

    Two examples 38

An experiment with JIGSAW-1 50

Appendix 64

    Diagrams of the test sentences 66

    The labels used in the experiment 81

    Syntax/semantics table for corpus 85

References 87

ABSTRACT

This paper reports a portion of a research effort to develop a program which will simulate the language learning behavior of humans. Here presented is a heuristic parsing procedure which accepts natural language sentences and produces for each a form of analysis called a "labeled dependency tree". The formal grammar on which the procedure is based differs from the "phrase structure" formalism of Chomsky (1957), and the analysis procedure attempts to discover the single most probable analysis rather than all analyses of ambiguous sentences.

Included are discussions of the syntax-meaning distinction, the special problems of simulation, and the need to handle a general class of inputs, and the need for analysis procedures which are to be self-organizing. The paper describes a computer program for analysis of sentences and reports an experiment with the program.

PREFACE

The author wishes to acknowledge the expert assistance of Lee Hafter and Jane Lindsay who wrote the test sentences, prepared the sentence diagrams, and compiled the inputs used in the experiment.

The research reported in this paper was supported in part by the Carnegie Corporation of New York and The University of Texas.

INTRODUCTION

The burgeoning interest in computer processing of natural languages has resulted in a large number of programs and proposed programs for producing parses of sentences (Bobrow, 1963). It is now clear that there is no single procedure best suited for all purposes, but rather the parsing procedure for a given task must depend upon the larger goals envisioned by the system designers. Thus a procedure which is appropriate for a natural language translation program will probably be unnecessarily elaborate for use in processing unambiguous computer languages as required in syntax directed compilers of the sort proposed by Irons (1961). Also, procedures based upon a "predictive analysis" scheme [Rhoades (1959), Lindsay (1961), Oettinger (1961)] are perhaps better suited for simulation of human cognitive processes than those based on phrase structure analysis [e.g., Pendergraft (1964)].

Research on automatic syntactic analysis has been greatly influenced by the work of Chomsky (1957,1959,1963) on formal descriptive linguistics. Although the program laid down by Chomsky is well-defined and intuitively appealing, the translation of Chomsky's work into a workable machine system encounters several difficulties. The most obvious difficulty is the discovery of the grammar for the natural language which the machine is to understand. One solution to this problem is to develop a learning procedure which will discover an appropriate grammar, or, equally as good, discover a parsing procedure which is appropriate to the language. The only extensive effort along these lines is the work of Knowlton (1962).

This paper reconsiders the problem of syntactic analysis and describes an approach which attempts to circumvent previously discovered difficulties. In particular, the procedure is directed toward developing a machine which learns to parse sentences in such a way that the resulting parse is usable for other processing.

CHOMSKY'S FORMALISM AND THE PROBLEMS IT RAISES

Consider a finite set, U, of symbols and the infinite set, S, of all strings formed by concatenating a finite number of instances of symbols from U. A language, L, is a subset of S, and a grammar, G, of L is a procedure which could produce all members of L but no strings which are not members of L. A parsing procedure is a program which will discover, for some strings of S if the string is in L, and may describe one or more sequences of steps such that if the sequence is performed by G the given string will be produced.

The above outline is the now familiar definition of syntax offered by Chomsky (1957). Also familiar are various metalanguages suggested by Chomsky and others for the representation of grammars. Perhaps the most frequently employed metalanguages are of the "phrase structure" type. These are described in detail elsewhere [see Chomsky (1959), Yngve (1961), and Backus (1959)].

The first difficulty encountered with Chomsky's analysis involves the phrase structure formalism. Chomsky states that phrase structure grammars are "quite neutral as between speaker and hearer" (1961, page 7). Although it is true that, given a phrase structure grammar, one can develop a parsing procedure which will produce all parses of any sentences of the language for the grammar, it does not follow that the parsing procedure is a convenient, efficient, or natural one. Indeed it may not be. Phrase structure grammars are a much more convenient formalism for defining a mechanism for producing sentences. For the purposes of analysis, predictive techniques seem to be more efficient, and as has been argued elsewhere, they are more appropriate if one is interested in simulating human language behavior (Lindsay (1963, Hockett (1961)).

A second difficulty raised by Chomsky's definition of the problem involves the discovery of a grammar for a language. In brief, no answers have been supplied to this problem. Current practice relegates the task of grammar discovery to a linguist or other person familiar with the language in question and the metalanguage of the grammar. Chomsky (1957) is pessimistic on this issue.

A third difficulty is that of defining the language to be studied. With formal languages this is usually accomplished by specifying the grammar, which thus defines the language. Chomsky proposes a similar procedure for natural languages: define a grammar which admits all sentences which are obviously in the language (e.g., "I am a man") and rejects all sentences which are obviously not in the language (e.g., "of skiggle the the, fard"); then use the grammar to decide on doubtful cases (e.g., "I is feelin' poorly").

It seems, however, that this suggestion, while making the scheme workable, does not really advance our knowledge of language. Intuitively, one would like to specify the language and then find a grammar for it. Unfortunately, deciding what is and what is not, for example, English, is an elusive project. Certainly, English is not a finite collection of utterences, and hence cannot be defined by reference to a corpus; we must include not only what has been said in English, but what might be said as well. But we must not include everything which might be said, because some of it will be said in French. And we cannot include everything which might be said by

English-speaking people, because some of them speak Russian as well.

Based on this discussion, Chomsky's position becomes more attractive. And yet if we accept this egress in order to get on with the business of developing automatic procedures for handling a useful range of natural language inputs, we may cut ourselves off from all hope of capturing the human's ability to process language.

This ability is noteworthy for its flexibility. For example, if a typographical error is encountered, a human does not indicate an error halt; in fact, it is almost always true that any two words can be transposed in a passage without disrupting processing, and even in cases where the sense of the passage is changed though not detected processing does not fail. If a strange dialect or idiom is encountered, or a novel turn of phrase is presented, most intelligent humans are able to make some sense of it. Indeed Chomsky's well worn example of a grammatical but meaningless sentence might pass for profound poetry in some circles: "colorless green ideas sleep furiously".

A system based on a fixed grammar and parsing procedure which must succeed before any further processing can begin will not be able to handle any of the above situations without being so extremely complicated as to be unusable.

Fourthly, since phrase structure grammars are so noncommittal on the question of analysis, the parsing procedure is not really defined by the grammar. If one is interested in obtaining all of the legitimate parses of an arbitrary input, there are well-defined procedures from which one may make a selection solely on the basis of efficiency. However it must be remembered that syntactic analyses are not the desired end result of language processing. Usually, seemingly syntactically ambiguous sentences are not ambiguous at all to the hearer. In fact introspection will usually reveal that the hearer is unaware of interpretations other than the single one he "has in mind". Further, if the hearer, while reading a sentence suddenly finds that things "do not fit" syntactically, he must begin again and search for the point of error; he does not simultaneously compute all possible parses. The minor problems of ambiguity thus far encountered with systems based on a rigid grammar and an exhaustive parsing procedure are illustrated by Oettinger and Kuno (1963). Their grammar and parsing procedure produces three parses of the sentence "They are flying planes". The least obvious is that in which "planes" is a verb. Our point is that this analysis is so obscure that most people fail to see it even after it is pointed out to them; clearly they did not produce it by themselves in any conscious, usable form. To handle this problem, a parsing procedure must have the property that it produces parses one at a time, the most probable one first and the correct one seldom later than second; and that the semantic rejection of false paths proceeds step-by-step (word-by-word) with the syntactic analysis.

Finally, Chomsky's program does not yield any insight on the important question of language learning. The formalisms thus far proposed are overly rigid in the sense that constructions are either legitimate or not legitimate; there is no provision made for assigning a probability to a construction (although it is readily possible to assign probabilities to members of a set of possible parses, each of which is legitimate). On the one hand this means that if one wishes to have a non-restrictive (overgeneral) grammar, it is necessary to specify more details of the parsing procedure lest it give equal credence to the unlikely parses and hence take an excessive amount of time to perform an exhaustive analysis. On the other hand it means that the information about the language is not divided by the grammar into portions which can be learned in a manner analogous to human learning: for example, the program either understands all infinitive constructions or none; it either understands the passive transformation or it does not. Although it is a gross over simplification to assume, as is the fashion in most current psychological learning theories, that learning occurs in tiny steps through some form of reinforcement, it is equally unlikely that the child learns powerful generalizationsin a single learning step. What is learned on a single trial lies somewhere in between, and neither phrase structure grammars nor transformational grammars appear to provide the right sized pieces for learning.

One approach to the problem of language (or at least grammar) learning is to simplify the grammar and complicate the parsing procedure. There are two avenues suggested by phrase structure formalisms. On the one hand, one could construct an undergeneral grammar which could produce no sentences not in the language but could not produce all sentences in the language.

The learning program would then add rules to the grammar as they are needed. On the other hand, one could begin with an overgeneral grammar which could produce all sentences in the language and many nonsentences. The learning program would then modify itself so as to discard analyses which were acceptable to the grammar but improbable for the language.

One difficulty with the first procedure resides in the fact mentioned above, that the pieces to be added are not the psychologically correct size. A second difficulty involves the manner in which the machine would be told to add a new rule. One possibility would be to present the program with a sentence involving a new structure and then present the analysis. It would then be a simple matter for the program to detect the new rule and add it to its grammar. This procedure is unsatisfactory because it is unrealistic. Children certainly are not taught in this manner and in fact could not be so taught until they possess a great deal of knowledge about language and about language analysis--something which most people never acquire. Furthermore, the burden on the teacher would be enormous. A more natural learning situation is one in which the teacher merely supplies more gross information, such as whether the child's reaction to (analysis of) the sentence is appropriate as a whole, without detailed feedback of the source of difficulty. It is desirable that a learning program be instructed in a similar manner.

The second procedure, employing an overgeneral grammar and a complex analysis procedure, appears to be more fruitful.

One such attempt has been undertaken by Knowlton (1962). The learning demonstrated with Knowlton's program was marked in the very early stages (the first 30 sentences), but quickly reached a fairly low asymptote (about 50% correct analyses). A sufficient number of variations in the program, however, remain to be tested, and hence the technique cannot be discarded. Perhaps, however, Knowlton's procedure of basing the decisions concerning the analysis path to follow on simple statistics collected from previous analysis is oversimplified.

The learning procedure to be employed in the present work is based on the second approach, although it differs from Knowlton's work in major respects. The details of the learning scheme will not be given in this paper. It will be apparent, however, that the linguistic description presented was developed with the learning problem in mind.

THE DISTINCTION BETWEEN SYNTAX AND MEANING

In light of the above, it would seem essential that, in order to decide on a syntactic processor, we have a formal definition of the subsequent processing. The general consensus of experts in the field is that the syntactic analysis procedure is fairly well understood, but that the problem of meaning has not been adequately formulated. And yet some would argue that, having shown the distinction between the three parses of "They are flying planes", a program has demonstrated some knowledge of meaning. Where does syntax end and meaningful interpretation begin?

Most persons admit to an intuitively felt distinction between the syntax and the semantics (denotative aspects of meaning) of a language. In formal logic, this distinction is clearly drawn, but in the study of natural language it is not. Thus one could argue that there exist meaningless but grammatical sentences (such as Chomsky's example: "Colorless green ideas sleep furiously") and that it is appropriate to capture this distinction through formal definition.

Chomsky has pointed out that his formal definition of grammar does not satisfy our intuitive notions of "syntax". For example, a listing of the sentences of L, if L is finite, is a satisfactory grammar by the definition. However, such a grammar is not useful because it is not "revealing".

Although "revealing" is not formally defined, Chomsky offers some examples. In the first place, he argues that a grammar *should* yield multiple methods of generating ambiguous sentences, as in the example of the flying planes and the planes which are being flown. Going even further, he argues for transformational grammars on the ground that they more readily reveal the common content of the two sentences "The man hits the ball" and "The ball was not hit by the man", even though the sentences are otherwise unsimilar.

The position adopted here is that grammars which go beyond a simple classification procedure (which decides whether a given string is in the language) are partial descriptions of meaning by virtue of the fact that they reveal structural relationships. While it is true that a formal definition of a language through a phrase structure syntax can be called a grammar, thus making a distinction between syntax and meaning, the distinction is arbitrary and could be drawn elsewhere by extending or restricting the set of grammar rules. Should "the man is the house with glasses" be parsed in more than one way? The decision is arbitrary.

It is possible to restrict the descriptive power of syntax to a procedure for classifying sentences without revealing structure by a technique other than a simple enumeration of sentences. This could be accomplished, for example, by a collection of rules which specify the legitimate local contexts of words of the language. Thus in a compiler, a few simple rules which admit the sequences "((" and "(a" but reject "==" and "+)" would constitute a grammar which defines the tactics of the language. The interpretation of an input such as "X = A*(B+C)" such that the essential structural relations involved are revealed would lie outside the domain of tactics, and outside the domain of syntactics if the grammar were so defined.

If one allows syntax to go beyond tactics, the point at which semantics enters has no intuitive currency. This is not to say that any other distinction is of no value; it is merely arbitrary. It is certainly feasible to define the grammar of a language to be a procedure which will generate all meaningful sentences of the language and no others (defining the meaningful sentences is no more difficult a problem than defining the language). Whether it is possible to write such grammars within any of the metalanguages thus far proposed is not known, but it seems unlikely.

REQUIREMENTS FOR A LANGUAGE PROCESSOR

Thompson (1963) has presented a cogent discussion of the concept of the structural interrelations in language. The view presented there, and adopted here, is that language serves to define the manner in which the (semantic) structures of the constituent referents are to be combined into a larger structure.

One sense of the word "dog" denotes to an adult English speaking person, a complexly structured entity, with hair, eyes, legs, teeth, bones, atoms, saliva, calcium, fleas, and other substructures interconnected in a complex manner. "Man" likewise denotes another complexly structured entity. The sentence "The dog bit the man on the leg" serves to interrelate the structures denoted by "man", "bit", "dog", and "leg" in such a way as to represent the event. In particular, those aspects of structure, which are of no importance such as the hair of the dog and the man's freckles, take no part in the operation.

This view of the function of language is in the tradition of presentational psychology and its descendant, Gestalt psychology. The process so described also lies at the heart of the problem solving theory of Duncker (1935), and is of importance in the development of inferential memories, as discussed by Lindsay (1964).

The theory of data structures is not sufficiently developed to enable us to program machines to build sophisticated representations of the full range of topics which may be discussed by a natural language. It is, however, possible to explore partially the manner in which such representations might be employed in a language processor. Thus it is possible to code interpretations of linguistic inputs in such a manner as to make use of the power of list structures while not discarding information which cannot be conveniently coded with these techniques. The residual information which cannot be placed implicitly in the associative structure may be carried explicitly by encoding with arbitrary symbols. (See Raphael (1964) for a discussion of this issue.)

Another desirable feature of a language processor is that it must not be halted by things it cannot understand, such as "%*#5/...;+=9)"; that it must extract whatever information is available from an input, such as the family relations described in "Fred's brother John fidge web in zot thethe and Ed married Fred whose Clyde is farny"; that it must develop reasonable interpretations of such inputs as "Colorless green ideas sleep furiously"; that it must develop reasonable hypotheses as to the reference of new words as in "The stick is plard"; and that it must be able to handle typographical errors and other word games in a reasonable manner, as in "The the saw tail dog man wag its".

Finally, it is perhaps true that attempts at developing learning programs have been overly optimistic in supposing that machines can learn complex things by random exposure to a sufficient amount of information. Children, or adults, seldom do. The process of teaching is an elaborate procedure which proceeds from the simple to the complex with plenty of practice along the way. Learning a language is one of the most complex tasks a computer will ever do, and it is not unreasonable to suppose that the education process will require an extensive and complex exposure. However, the machine should be resistant to bad teaching, just as children are, so that poor instruction over short periods of time can be overcome without permanent impairment. In summary, we require as a minimum the following:

1. A language processor should build list structures which preserve and encode all the information necessary for construction of the interpretation when we discover how to represent it.

2. A language processor should accept any input consisting of strings of symbols from the alphabet of the language and should process the input, interpreting all of it or as much of it as an intelligent adult could interpret.

3. A language processor should be able to learn a language through a judicious training sequence no more extensive (and probably no less extensive) than that employed in teaching a child to speak and read.

JIGSAW

A proposed computer program called JIGSAW has been designed to meet the criteria outlined in the previous sections. It will described here, and a later section will describe in detail the first version of this program, JIGSAW-1, which is operational. JIGSAW-1 does not embody all of the features proposed for the JIGSAW system.

The basic organization of JIGSAW [see Figure 1] embodies a procedure which is basic to heuristic programming: the system is divided into two sections, one which proposes a line of attack, and a second which attempts to carry out the proposal. In the context of parsing programs, this principle has, according to Hays (1961, page 368) "been invented, lost, and reinvented many times". In the program described by Hays, the procedure divides syntactic rules into "word-order rules" to propose constructions and "agreement rules" to test the proposed construction. JIGSAW employs a similar division, the details of which will now be described.

Strings formed by the concatenation of symbols (words) are input and an interpretation of the string is constructed. The interpretation is a structural representation of the relationships among the words, and is expressed as a graph of some or all of the words. A dictionary specifies the possible structural interconnections. A syntax directory is used to retrieve syntax routines which select from the set of possible interconnections those which are to be attempted in the given linguistic context. If a proposed connection is possible according to the dictionary, it is effected; if it is not possible, it is abandoned. Processing proceeds in a single left-to-right scan of the input string and produces a single result; however, the result may consist of several disjointed structures. If the result is rejected by an external agency called the teacher, JIGSAW is able to construct a second result. If JIGSAW's cognitive limits are exceeded during the processing of an input, it is possible to force the program to combine separate pieces of its structure. This may be done by (a) forcing a structural connection which is proposed by a syntax routine even though it is not acceptable to the dictionary,

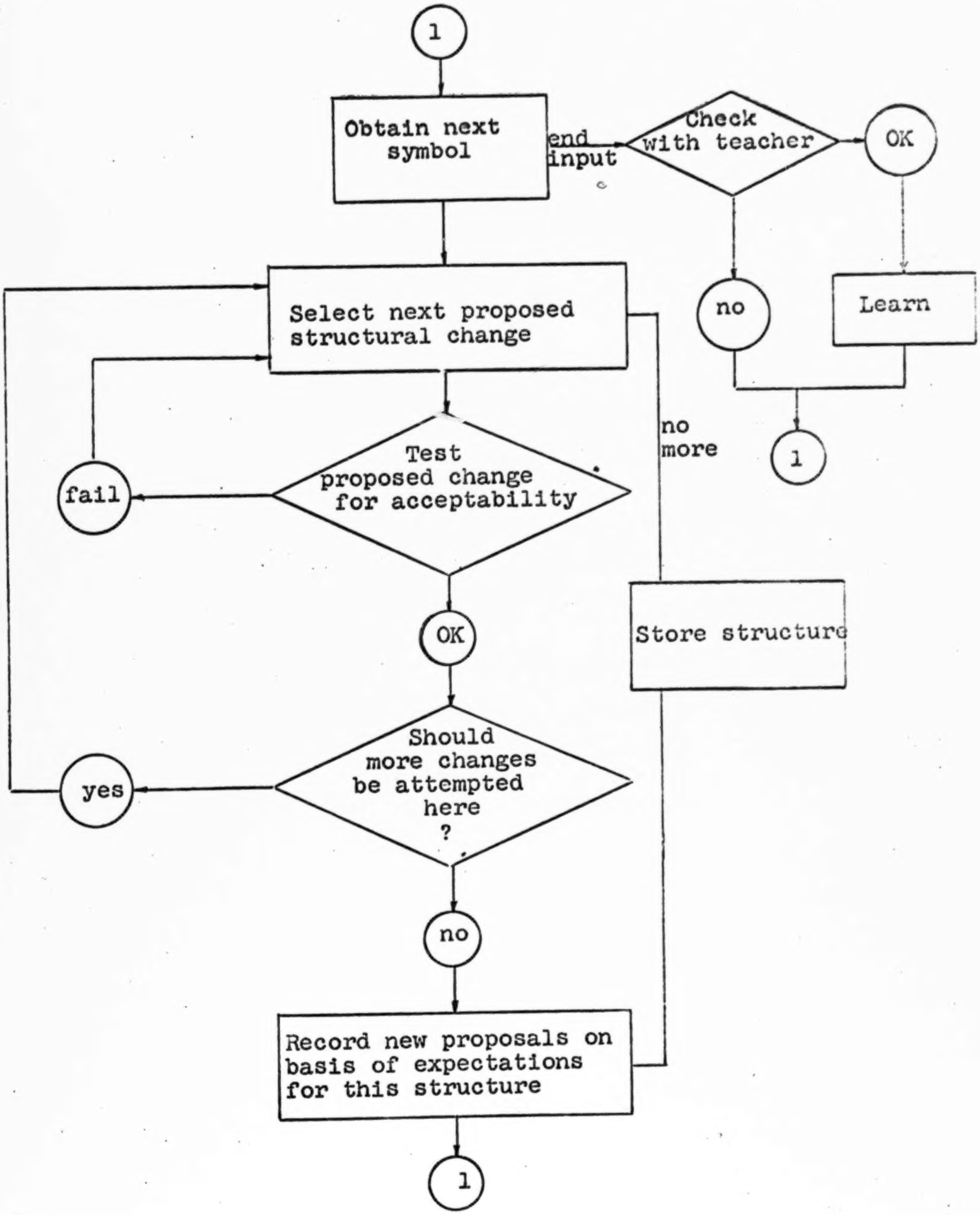

1
Obtain next symbol
end input
Check with teacher
OK
no
Learn
1
Select next proposed structural change
no more
Test proposed change for acceptability
fail
OK
Store structure
Should more changes be attempted here ?
yes
no
Record new proposals on basis of expectations for this structure
1

Figure 1. Main Flow of JIGSAW

or (b) employing a structural connection which is acceptable to the dictionary even though it has not been proposed by any syntax routine. If any structures so formed are acceptable to the teacher, JIGSAW may learn by modifying its dictionary or syntax directory so that the structure forced is now acceptable, and/or by changing a syntax routine so that the structure is in the future proposed in similar linguistic contexts.

The behavior of JIGSAW may be described for purposes of illustration by the following simple example. Assume that the dictionary allows structural connections of type F1 and F2 to exist from "red" to "rose", and structural connections of type F1 and F3 to exist from "rose" to "red". Assume also that when "red" is encountered, a syntax routine retrieved from the syntax directory proposes a structural connection of type F1, and sets up an expectation for something which will fulfill this proposal. When "rose" is next encountered, an interrogation of the dictionary reveals that it is possible to form the F1 connection from "red" to "rose", so this is done. If the opposite sequence "rose red" had been encountered, a similar procedure would have produced the similar but reversed structure, with "red" acting as a noun. If the teacher indicates that this is acceptable, no change is made in the dictionary or the syntax tree.

If a new input string begins with "red" and is followed by "clyde", the proposed connection is rejected by the dictionary which has no information about "clyde"; the two words are saved as separate structures. If, however, saving the two words exceeds the cognitive limits, a connection of type F1 is forced, since that is the only proposed connection. If the teacher accepts the resulting structure, the dictionary is modified to make this connection a possibility in the future. If now the next input string begins with "farny", about which no information exists in the dictionary and for which no syntax routine is recovered from the syntax directory "farny" is stored. Assume the next word to be "rose". No connection is proposed, so the second word is saved. If this action exceeds the cognitive limits, the dictionary is searched for a possible connection; an F1 connection from "farny" to "rose" is established. If this is acceptable to the teacher, the syntax directory is modified to include a routine for "farny" which will propose an F1 connection in the linguistic context of "farny". A fourth string, "farny clyde" will now be processed to produce a structure relating "farny" to "clyde" with an F1 connection, even though JIGSAW originally had no information about either word.

Brief descriptions of each of the features of JIGSAW will now be given to indicate the range of possibilities envisioned. In the next section, a particular interpretation for each feature will be described in the manner in which it has been implemented in an operational version called JIGSAW-1.

The interpretation. The result of processing a string should be a representation of the "meaning" of the string. Unfortunately, we do not know how to represent meanings, although it is possible to specify some criteria for the representation. This has been done by Lindsay (1961), where it was pointed out that it is necessary for the representation to model the subject under discussion in such a way that the essential relations among the concepts discussed are reflected by the representation. This would allow inferences to be made beyond the information explicitly contained in the input, and would also allow the machine to perform "gedanken experiments" to test its understanding of the subject. Lindsay (1961) has programmed a linguistic machine to construct such representations for a very limited class of subject matter; Raphael (1964) has explored a more general but less powerful representation in the SIR program, but avoided the problem of translating from natural language to the representation.

Simon (1962) in programming the heuristic compiler has explored the problem of translating from natural language to a representation in the form of description lists, which provide a representational language of sufficient power to handle certain expressions which might be encountered in an imperative language of the kind which might be employed as a programming language. Essentially the same ideas have been extended and elaborated by Thompson (1963). Thompson argues that, since we do not have programming languages or theoretical expertise to describe and deal with arbitrary data structures, it would be fruitful to attack the problem of translating from natural language into representations using the data structures for which current techniques are available. Although this limits the subject matter which can be handled, the range is still quite broad in relation to the present state-of-the-art. Thus, in his DEACON, system, Thompson attempts to translate from natural language into data structures which can be represented with list structures, description lists, arrays, and numbers, since techniques exist for dealing with these structures. It is probably true that these structures are not powerful enough to represent systems which change with time in a continuous fashion, nor to deal with declarative sentences and the consequent inferences which they entail. Nonetheless, question-answering systems of significant versatility can be defined with these techniques. Thus in response to the question "What men in battalion 10 have served for more than 20 years without an accident?", DEACON might construct a list of men satisfying the requirements. This would be done by consulting a data store which represents men, battalions, years of service, and accident records with list structures. The most important feature of DEACON is its ability to retrieve the requested information from any of a variety of explicit data representations; that is, the data may be organized as a list of men with associated accident records, service records, and battalion assignments, or as a list of battalions with associated personal files, or any of several other representations. The basis of DEACON is a set of rules which prescribe transformations from linguistic inputs to routines which modify data structures composed of lists, etc.

JIGSAW envisions using structures more general than list structures. Although no language exists for defining or processing these structures certain design specifications have been outlined and will be described in a later paper [cf. Lindsay, Dauwalder, Pratt, & Shavor (1964)]. Data structures allowable in this system will be (a) arrays and (b) any structure which is isomorphic to a colored line graph composed of nodes connected by labeled (colored) lines. Within this framework, limitations may be imposed by any particular version of JIGSAW. Such limitations serve as a useful heuristic device to check the legality of proposed partial structures. In JIGSAW-1 (see below) structures are limited more severely.

Although it is not clear that colored graphs will allow the automatic inference properties discussed above, such graphs at least allow the preservation of the necessary information. Thus we might wish to represent the statement that event Z follows event Y. Although list structures do not permit this in a "natural" way, a special symbol denoting "follows in time" may be used as an attribute on the description list of Y and be given the value Z. JIGSAW at present merely aims to construct such information preserving representations and does not attack the more difficult problem of constructing good representations. Clearly this limited goal can largely be achieved within the confines of list processing.

The dictionary. In order to allow the overgeneral linguistic description which is necessary to handle an almost unrestricted class of inputs, the possible structural connections are stored with few references to macroscopic structural features. Thus, most structural information is stored on a word-pair basis; the possible ways in which two words may be related do not depend upon the other connections entered into by the words. For example, the dictionary might indicate that a connection is possible between "tall" and "man" without reference to any other aspects of the phrase in which "man" occurs. This would make possible a sentence such as "the short, tall man". On the other hand, the facility for recording phrase dependent structural rules is available when needed. Thus it would be possible to allow a connection between "even" and "number" while disallowing a connection between "even" and "prime number".

The information contained in the dictionary is of importance not only for the analysis process, but for learning as well. It is desirable to have available a test of consistency which could be employed to check new dictionary entries suggested by a learning procedure. If the suggested information is inconsistent with other information in the dictionary, it would be rejected. The work of Sommers (1961) suggests a possible test of consistency. Sommers defines a predicate "←" with the interpretation that Q ← P if of what is P, it can sensibly be said that it is Q. Under certain assumptions, he is able to show that meaning classes may be arranged into a tree. If Sommers' analysis is accepted, a suggested sense relation could be tested for consistency with the requirement that the "semantic tree" be maintained.

The syntax directory and syntax routines. A scheme for generating proposed connections is needed. A syntax routine is merely a program which assigns to a partial interpretation a subset of the possible further connections into which the structure may enter. The syntax directory is the store for syntax routines together with a procedure for retrieving them as a function of the linguistic context in which the partial interpretation is imbedded. Similarity of linguistic contexts is defined by the retrieval rules. Thus to select a syntax routine, certain questions about the input are asked [e.g. "Is a noun phrase present?", "Is the word "to" present?" "Is there any punctuation?", "How complex is the structure so far?"]. The answers to these questions determine which syntax routine will be selected.

Certain features of a linguistic utterance denote the concepts being discussed and other features specify how the structures associated with these concepts are to be combined. Thus in the phrase "red rose", the concepts discussed are redness and roses and the relation connecting the concept is that "*the color of* the rose is red". The connection is determined by the cue of *word order*.

Typically, syntactic analysis attempts to interrelate all the words in a sentence on the basis of positional and punctuation cues. Thompson (1963) has made the important observation that the usual dichotomy which places words in one category and positional cues and punctuation in another category, is misleading. A more revealing dichotomy is between *referent words*, which denote concepts with structure, and *function words*, punctuation, and positional cues, which give information as to how the structures of the concepts denoted are to be combined. The identification of function words must, of course, be formally specified by a linguistic description. However, the class would include most conjunctions, prepositions, articles, and logical connectives.

In JIGSAW, the distinction between referent words and function words is sharply drawn but not easy to identify. Any word about which structural connection information is given in the dictionary is a referent word; any word whose presence or position is used as the basis of a retrieval function of the syntax directory is a function word. However, it is possible that the same grapheme may act as either a referent word or as a function word in the same or different linguistic contexts. There is not a simple way to decide which role a given word plays in a given context without a detailed examination of the dictionary, the syntax directory, and the analysis procedure. It is important, however, to note that the interpretation of a string may not explicitly contain all of the words from the string; the information conveyed by some words may be translated into structual features of the representation, in the same manner as is done with positional cues.

The teacher. The acceptability of an interpretation is decided by an agent external to JIGSAW. In actuality, the decision is made by a human who examines the program's performance. The teacher, however, simply approves or disapproves, and is not able to point out the sources of difficulty. This strategy assures that the basis of JISGAW's learning resides within its linguistic abilities. The programmer may teach JIGSAW a specific piece of information only through an appropriate selection of linguistic examples.

The learning of language is accomplished by comparing linguistic examples with reality. In human behavior this may be accomplished by simply receiving approval or disapproval for a response to a sentence supplied by a teacher. However, it may also be accomplished by producing linguistic behavior which is judged by asking specific questions of the teacher, or by asking questions of the environment in other ways, such as by performing experiments (e.g., touching a stove after the teacher has said "the stove is hot" comprises an experiment which provides information about the correspondence of language and reality). JIGSAW's limited ability to interact with the environment will prove to be a real limitation upon its learning ability. Nonetheless, the limited information obtainable from the teacher should be sufficient for some interesting learning, if the program is able to use it properly.

Cognitive limits. Any sort of condition is permissible as a cognitive limit. The effect of exceeding a limit is to force JIGSAW into a second, more desparate mode of behavior which attempts to make structural decisions normally put off until more information is available. A variety of conditions may act as limits. For example, exceeding a limit on processing time or upon the amount of memory used for a given analysis, or any arbitrary limit imposed at random or regular intervals. After each word is processed JIGSAW employs a test which determines whether any limit has been exceeded.

JIGSAW-1

A program, coded in IPL-V [see Newell, et al. (1964)], has been written to test the feasibility of the JIGSAW approach. This program does not contain the full flexibility discussed above, but provision has been made for adding all of the described features. A detailed documentation of JIGSAW-1, including a listing of the program, is available in a separate paper [see Lindsay (1964)].

For JIGSAW-1, the interpretation is a labeled dependency tree, the dictionary is a simple table composed of IPL description lists, the syntax directory is a tree patterned along the lines of EPAM as described by Feigenbaum (1959), the cognitive limits take the form of a parameter controlled limit on the number of separate structures which may be held in memory, and the learning program is non-existent.

Labeled dependency trees. A syntactic analysis closely related to immediate constituent analysis is dependency analysis, first proposed by Hays and Ziehe (1959). This formalism displays structural relations among the words of a sentence by means of a tree which has words at the nodes and directed line segments connecting the words. [See Figure 2] The word at the tail end of the arrow is said to "depend" upon the word at the head of the arrow. The relation of dependency may intuitively be thought of as follows: removal of the governing word would make the dependent words meaningless, while the dependent words may be removed or substituted for without destroying the sense of the sentence. Modifiers generally depend upon that which is modified; subjects depend on verbs; objects depend on their prepositions; etc. Formally, the dependency relations are defined by a grammar, but they are meant to reflect the structures of the language, as are phrase structure rules.

A labeled dependency tree is a dependency tree in which the connecting lines bear labels which reflect the sense of the connection. Again, the placement of these labels is formally defined by the grammar, but the intention is to reflect the sort of structural interconnection between words which the sentence is meant to convey. Thus "green" might depend on "tree" in the same sense (via the same label) as "white" depends on "snow", but in a

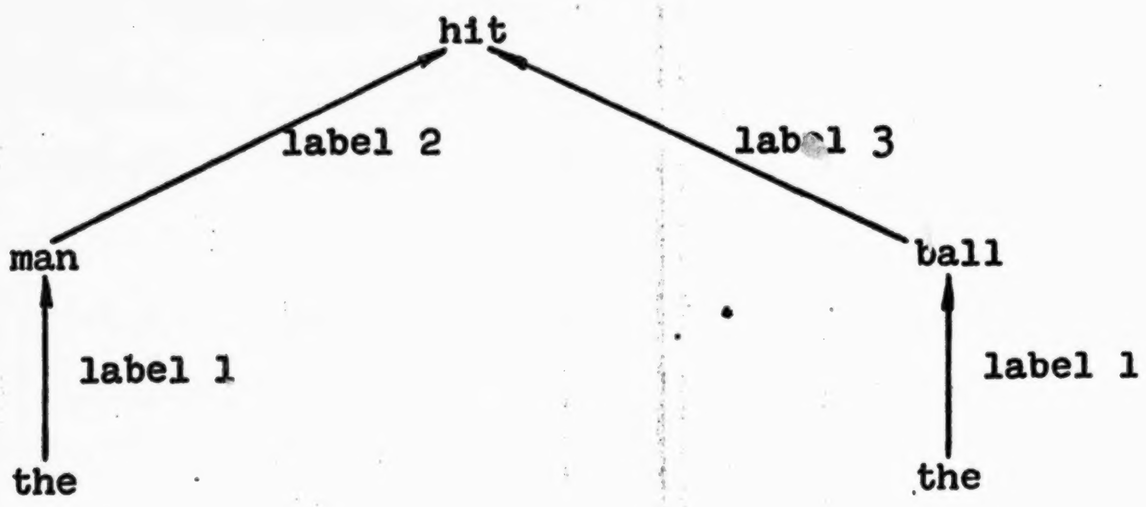

hit
label 2
label 3
man
ball
label 1
label 1
the
the

Figure 2. A Labeled Dependency Tree

different sense than "green" depends on "recruit", even though in all cases the dependency reflects modification in the usual syntactic sense. In most instances, "the" would be dependent upon a noun via the same label; but in some instances, such as in "the more the merrier", the label might differ.

The requirement that the interpretation be a tree is a marked restriction. It is, however, a useful restriction, since it serves to reduce the number of possible structures and hence make the search easier. Thus, when a proposed connection is tested with the "Test for possibility", it is rejected if making the connection would violate the tree constraint by introducing loops, even though the dictionary finds the connection acceptable.

If we consider interpretations which are graphs restricted only in that at most one label of a given type may exist in each direction between any two word instances, then the number of possible structures, assuming no constraints imposed by the dictionary, is $\sum_{K=1}^{LN(N-1)} \binom{LN(N-1)}{K}$, where L is the number of labels and N the number of word tokens in the sentence. If the dictionary allows on the average five labels per word (as in the experiment reported later) the number of possible combinations is on the order of 10^{120}. If we require that the structure be a tree, so that each word has at most one governing word, the number of possible structures is reduced to $N!L^{(N-1)}2^{(N-2)}$. With $L = 5$ and $N = 10$, the number of possible trees is on the order of 10^{11}.

The dictionary. In this program, no distinction is made between function words and referent words. All words in an input string are found in the dictionary, and all occur in the resulting interpretation. To store information about the possible structural connections between each pair of words would require an excessive amount of storage space. Instead, each word has associated information which specifies which labels may be employed in dependency relations involving the given word. In particular, each word has two associated lists. List D3 for word W contains all labels which may be attached to arrows with W at the tail (dependent) end; list D4 for word W contains all labels which may be attached to arrows with W at the head (governing) end. In testing to determine whether a connection involving label X may be made between words W and Y with W being dependent on Y, a positive answer is made only if X occurs on the D3 list of W and on the D4 list of Y. If X is present on both lists the "Test for possibility" yields the answer "Possible"; if X is absent from one or both of the lists, the test yields the answer "Impossible", unless one of the lists involved is non-existent (indicating an unknown word), in which case the answer is "Unknown". As noted above, a connection must also satisfy certain structural requirements in order to be acceptable.

It should be noted that the above manner of storing information is less restrictive than would be the case if information were stored on the basis of word pairs. Thus the dictionary writer might wish to state that "green" may depend on "grass" via the label "color" and the "white" may depend on "snow" by the label "color". Using the above scheme, the resulting dictionary would allow "green snow" and "white grass". If this possibility is to be ruled out, two color labels must be used.

The syntax directory. In JIGSAW-1, syntax routines are stored at the terminal nodes of a tree. The non-terminal nodes of the tree have associated tests. These tests may be any routines which produce single symbols as output; in fact they are selected from a list of tests supplied by the programmer and are not generated by JIGSAW-1. Typically, the tests ask questions about partial interpretations and the answers to the questions direct the program to the syntax routine which defines the types of further structural connections which the program will propose for the disposal of the partial interpretation which was used as the entry to the tree. Typical tests are: "Is the interpretation a single word or a more complex structure?" (answer: simple, complex); "What is the word which is at the head of the interpretation?" (answer: a word type); "What is the label which is attached to the first level connection?" (answer: a label name); "How deep is the interpretation?" (answer: an integer).

When a partial interpretation is submitted to the syntax tree, the first question is asked of it. The result of this test selects the branch which leads to another test, and so forth, until a syntax routine is retrieved. Executing the syntax routine selects the appropriate proposals and attaches this information to the partial interpretation where it is carried for later use. In the experiment reported later the tests in the syntax tree ask only about the head of the interpretation and not about substructure.

This method of information retrieval is patterned after the EPAM theory of verbal learning as developed by Feigenbaum (1955).

The major feature of such a storage device is that the sequence of tests which defines the retrieval properties is stored as a data structure which may be dynamically altered by the program. Typically the alterations are effected in order to adapt to a particular retrieval task with which the program is momentarily faced. Thus changes in the tree may bring about unexpected consequences which are felt when the program faces a new task. The tree thus brings about discrimination and generalization learning. In the context of human rote memorization tasks, the EPAM theory has demonstrated learning behavior remarkably similar to that displayed by humans as reported in the extensive literature on human verbal learning.

In JIGSAW-1, which has no learning capabilities, the full vocabulary and associated syntax routines are given to the program at the outset. A subroutine grows a syntax tree in such a manner that each word is sorted to an unique terminal at which is placed its syntax routine. The algorithm which grows the tree sorts the given word in the tree. If an unique terminal is found, the routine is finished. If not, an attempt is made to find, from the given set of tests, a test which when added will discriminate the given word from the others with which it is confused. This is always possible due to the way in which the tests are defined. However, the list of tests is ordered from general to specific so that the algorithm examines general features of the structure first.

The proposals which are selected and marked by a syntax routine are similar in content to the information found in the dictionary. A proposal is of the form "Set up an expectation that subsequently a structure (perhaps a single word) will be encountered which can be combined with the present structure in such a manner that the present structure is dependent (governs) the expected structure by way of the label X". A single syntax routine may mark any number of such proposals.

Cognitive limits. The limitations placed upon JIGSAW-1 are that the number of partial interpretations must not exceed a small number (which is parameter controlled, usually set at seven) and that an attempt should be made to construct a single connected interpretation for a single input string. As indicated below, if a limit is exceeded, the program attempts to force a connection which has not been proposed or which is not compatible with the dictionary. Such attempts may fail, in which case the program gives up.

Initializing the program. The program may be initially supplied with any amount of information about the language to be processed. Some information must be supplied if the program is to perform other than at random. If no information involves distinguishing partial interpretations on the basis of their substructures, the program may be initialized by a simple algorithm, which has been employed in the experiment described in this paper. The algorithm for initialization proceeds as follows.

Select a corpus of sentences (or other units) from the language. Using your linguistic knowledge and intuition, draw a labeled dependency tree for each sentence, supplying whatever labels necessary. For each word in the vocabulary note all sites where the word occurs at the tail of an arrow. Form a list (the D3 list) of all of the labels associated with such sites. Each label is to occur on the list only once in spite of multiple occurences in the labeled dependency trees; the order of the labels on the lists is arbitrary. Next note all sites where the word occurs at the head of an arrow and form the D4 list in a similar manner. Add this information to the dictionary in association with the word.

Next consider each entry on the D3 list in turn. Determine whether, in any instance where the associated site occurs, the given word precedes (in the input sentence) by one or more word positions the word which appears at the other end of the dependency relation. If so, add the label to a list called the Q101 list, and consider the next D3 entry. If not, skip the label and consider the next D3 entry. When the D3 list is exhausted, consider the D4 list in like manner, this time forming another list, the Q100 list. Associate lists Q100 and Q101 with the syntax routine which is to correspond to the given word. Enter the word and its syntax routine into the syntax tree by means of the tree growing subprogram.

Program outline. After the syntax tree is grown and the dictionary read into memory, control is transferred to the main executive, E2, whose flowchart appears in Figure 3. E2 selects the next sentence to be processed and sequentially presents each word in the sentence to the main subroutine, E20:PROCESS CURRENT PHRASE. E20 either combines the current word with structures previously held in the immediate memory (IM) or reports failures after retrieving and executing its associated syntax routine. Routine E23 is executed to add the word to the IM and, if IM is

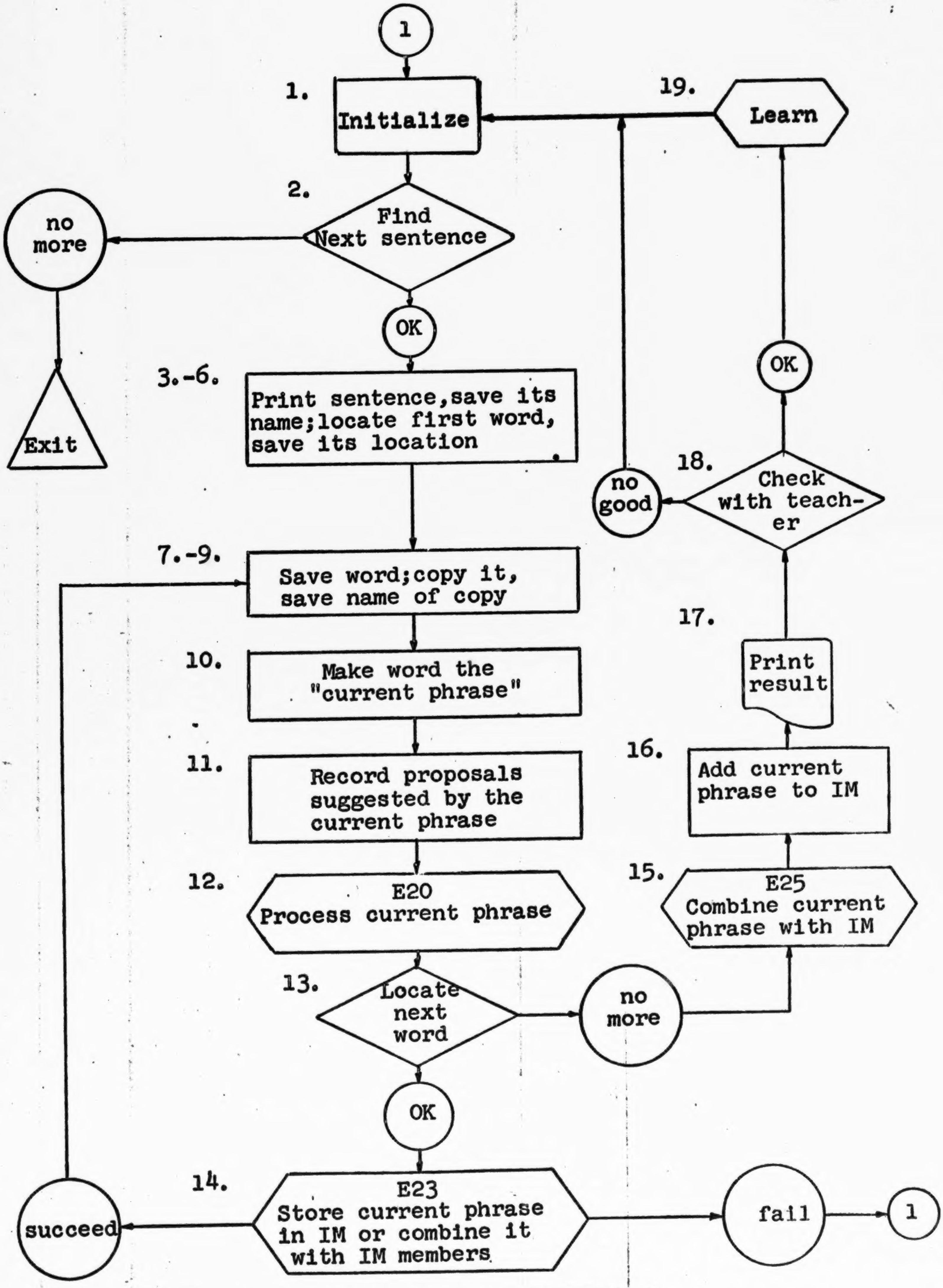

1
1.
Initialize
19.
Learn
2.
Find Next sentence
no more
Exit
OK
3.-6.
Print sentence, save its name; locate first word, save its location
OK
18.
Check with teacher
no good
7.-9.
Save word; copy it, save name of copy
17.
Print result
10.
Make word the "current phrase"
16.
Add current phrase to IM
11.
Record proposals suggested by the current phrase
12.
E20 Process current phrase
15.
E25 Combine current phrase with IM
13.
Locate next word
no more
OK
14.
E23 Store current phrase in IM or combine it with IM members
succeed
fail
1

Figure 3. E2, Main Flow of JIGSAW-1

full, to force a connection. When the sentence is completed, routine E25 attempts to combine the members of IM into a single structure. The result is then submitted to the teacher, T20, and if it is acceptable, the learning routine, E26, is executed. Then the next input is selected.

The heart of JIGSAW-1 is E20 and its three associated subroutines, G5, E21, and E24. G5 generates (in the IPL sense) each proposal associated with the phrases held in IM. The order in which generation occurs is parameter controlled. In the experiment reported, the IM is examined from most recent to most remote phrase, and for each phrase, each proposal which would use the current phrase as a governor is selected in turn. When these proposals are exhausted, the IM is again scanned in the same direction and proposals are generated which would make the current phrase dependent. Each proposal is presented to one of the subroutines E24 or E21, whose flowcharts appear in Figures 4 and 5. The subroutine may either terminate the generation of proposals or request the next proposal. As seen in Figures 4 and 5, generation is continued if the proposal is rejected or if the proposal is used to employ the current phrase as a governor. If a proposal which employs the current phrase as a dependent is used, the current phrase becomes part of another structure which is stored in the IM. The new structure is sorted in the syntax tree to retrieve a new set of proposals, and generation is terminated, thus returning control to E2 which selects the next word.

E24 checks the feasibility of the proposed structural changes by two tests. The first checks to see that no rules of

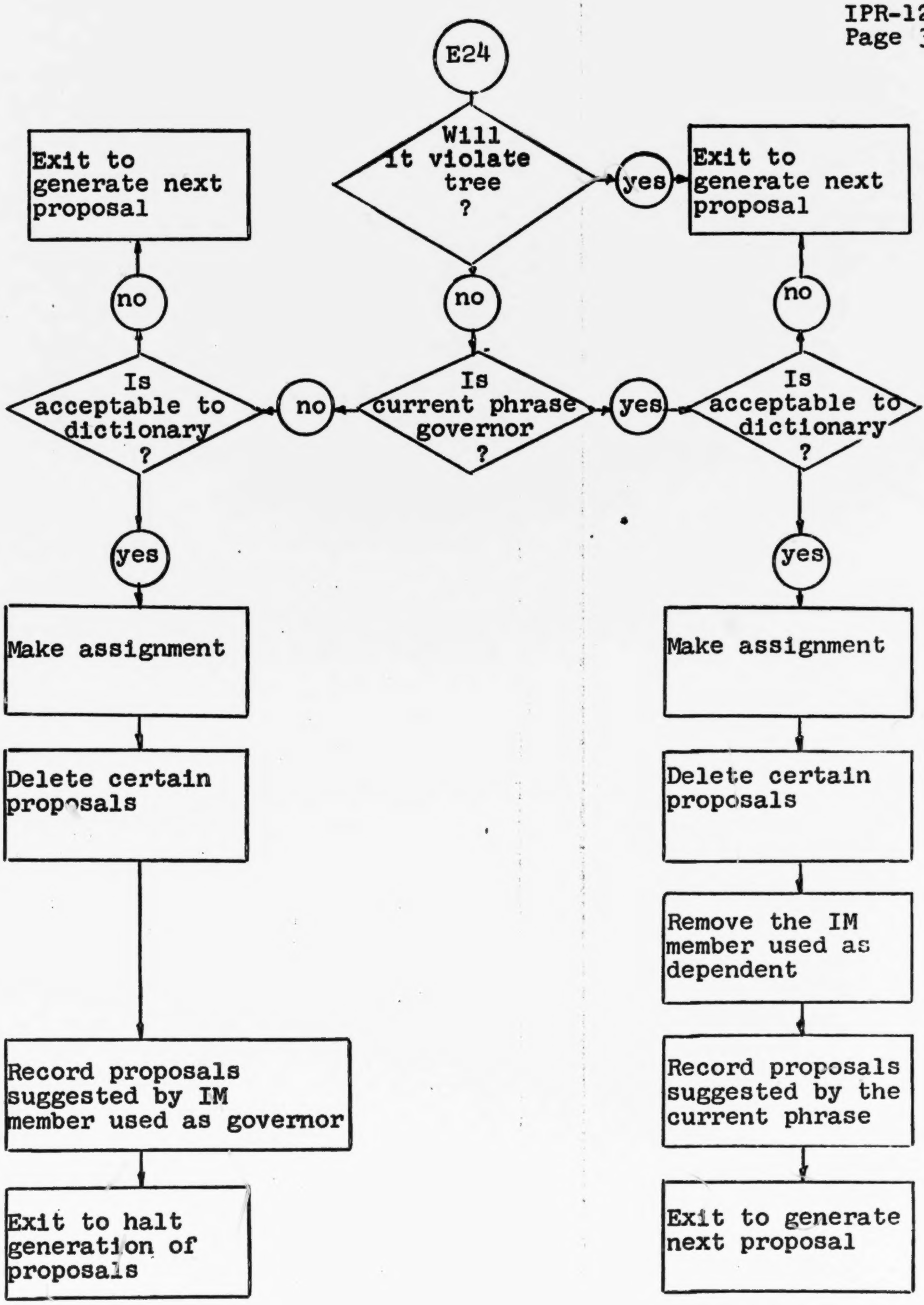

E24
Will it violate tree ?
yes
Exit to generate next proposal
no
Is current phrase governor ?
no
Is acceptable to dictionary ?
no
Exit to generate next proposal
yes
Make assignment
Delete certain proposals
Record proposals suggested by IM member used as governor
Exit to halt generation of proposals
yes
Is acceptable to dictionary ?
no
Exit to generate next proposal
yes
Make assignment
Delete certain proposals
Remove the IM member used as dependent
Record proposals suggested by the current phrase
Exit to generate next proposal

Figure 4. E24, Process Proposal

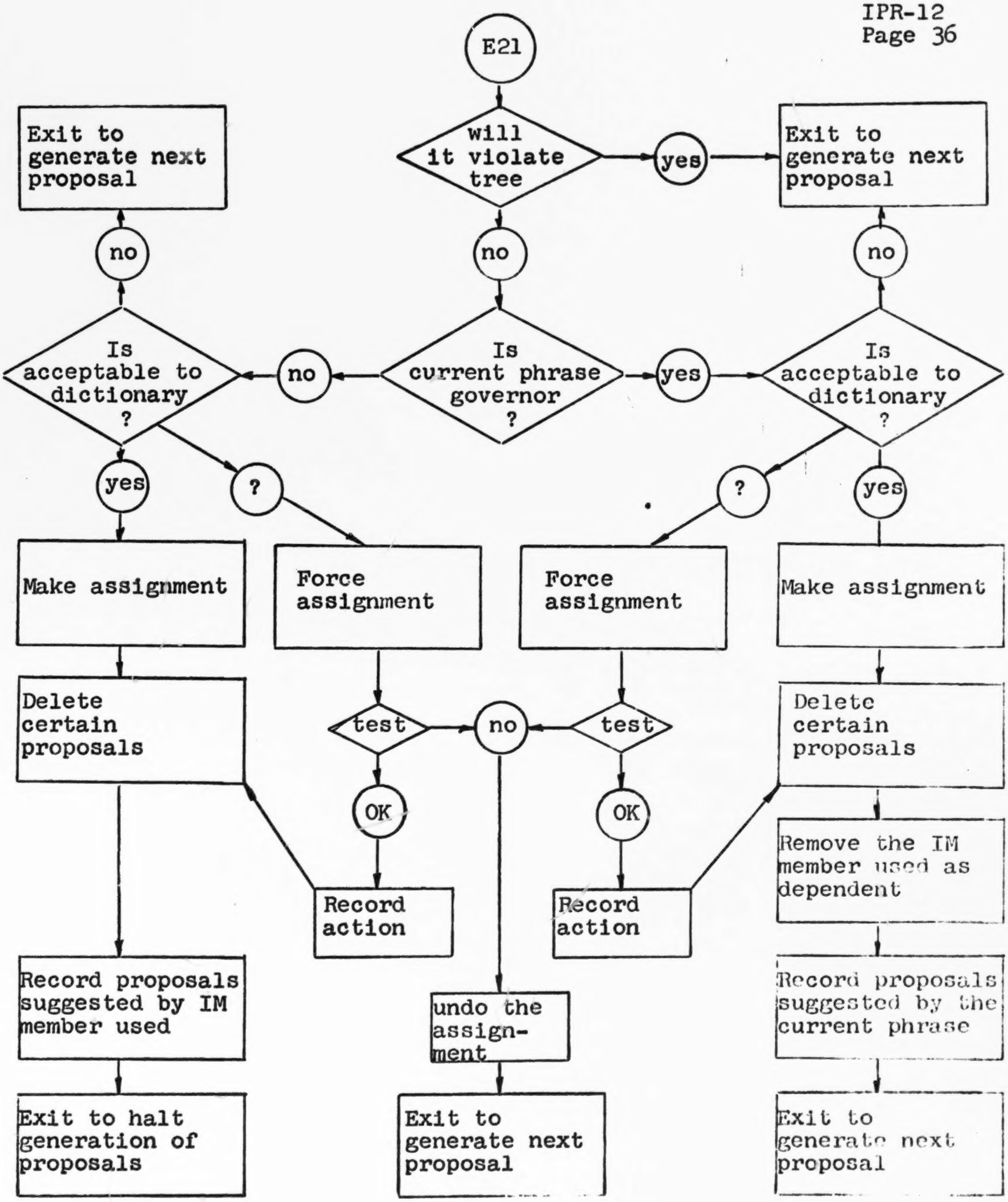

E21
will it violate tree
yes
Exit to generate next proposal
no
Is current phrase governor ?
no
yes
Is acceptable to dictionary ?
Is acceptable to dictionary ?
no
no
Exit to generate next proposal
yes
?
?
yes
Make assignment
Force assignment
Force assignment
Make assignment
Delete certain proposals
test
no
test
Delete certain proposals
OK
OK
Record action
Record action
Remove the IM member used as dependent
Record proposals suggested by IM member used
undo the assignment
Record proposals suggested by the current phrase
Exit to halt generation of proposals
Exit to generate next proposal
Exit to generate next proposal

Figure 5. E21, Process Proposal

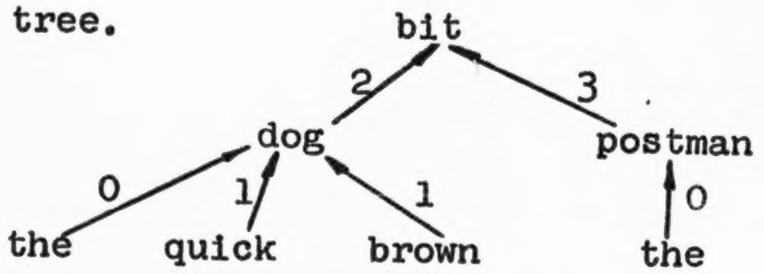
structure formation will be violated (in this case, the rule that the proposed structure must be a tree) and the second checks the proposal for consistency with the dictionary. If both tests are passed, the change is carried out.

If a change in structure is accomplished by E24, E20 is finished. However, if no change is made by E24, JIGSAW-1 makes a more concerted effort to accomplish something by regenerating the proposals and executing a more lenient analysis, E21. Even though a proposal is not clearly acceptable to the dictionary, E21 will force the proposed restructuring if it is not definitely rejected by the dictionary. This occurs at present only in the case where one of the lists, D3 or D4, for one or both of the words, is empty, indicating that the dictionary has no information concerning the relevant aspects of the words involved.

An additional complexity should be mentioned. A phrase is stored in the IM by placing the name of the tree in a memory register. A proposal may suggest that the word at the head of the structure is to be combined as a dependent or governor, or it may suggest that one of the words already dependent on the head of the phrase is to be used as a governor of the current phrase. Information concerning each proposal is formed into a list, and the list of proposals is associated with the name of the phrase. Since the generator, G5, produces proposals in the order in which they occur on the list, their placement is crucial. Implicit in JIGSAW-1 is the assumption that the language being processed is predominately an immediate constituent language. Thus, the order of the proposals is constructed so that proposals involving more recent words are placed nearer the top of the list of proposals, whether or not the words are near the top or bottom of the phrase whose name appears in the IM.

Two examples. JIGSAW-1 produces a trace at the level of description presented in the flowcharts. The trace and the results of the processing of two sentences are produced in Figure 6. In conjunction with the flowcharts, the trace is self-explanatory. The output encodes the labeled dependency trees in a functional notation. Associated with each word is a pair of parentheses which enclose the branches of the tree which immediately proceed from the word. Each branch is identified by a code of the form "Fxx--A()--B()etc." The label is denoted by Fxx (the letter 'F' followed by a decimal integer) and A, B, etc. are the words at the ends of the branches so labeled. Within the list of branches associated with the word, the branches with distinct labels are separated by commas. The following example illustrates this notation:

labeled dependency tree.



functional code for the above tree.

```
bit(F2--dog(F0--the(),F1--quick()--brown()),F3--postman
(F0--the()))
```

The "correct" interpretations of the two example sentences are depicted in the Appendix, where they are labeled, "sentence 0" and "sentence 1". It will be seen that sentence 0 was correctly

```
CURRENT SENTENCE IS....
THE      COMMITTEE      DECIDED      TO      TABLE      THE      BILL      UNTIL
FURTHER      NOTICE      .
CURRENT WORD IS...THE
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E21
Q23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
CURRENT WORD IS...COMMITTEE
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =THE  ,ATT =F70,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =THE  ,ATT =F,CURRENT TO BE SYMB
    *POSSIBLE.ASSIGN -THE- TO -COMMITTEE- VIA -F
Q07. MAKE 1X50 A VALUE OF 1X60 ON 1X70.
Q12. REMOVE NEED OF SYMBOL FOR 1X60 ON 1X50.
Q06. REMOVE NEED OF SYMBOL FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q36. REMOVE NEED FOR ALL SYMBOLS FOR 1X50 AS INDICATED ON 1X51.
Q13. DELETE SUPERPHRASE FROM IM, MOST RECENT OCCURRENCE ONLY.
Q22. DO SYNTAX OF CURRENT.
Q23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
CURRENT WORD IS...DECIDED
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =COMMITTEE  ,ATT =F72,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =COMMITTEE  ,ATT =F4,CURRENT TO BE SYMB
    *POSSIBLE.ASSIGN -COMMITTEE- TO -DECIDED- VIA -F4
Q07. MAKE 1X50 A VALUE OF 1X60 ON 1X70.
Q12. REMOVE NEED OF SYMBOL FOR 1X60 ON 1X50.
Q06. REMOVE NEED OF SYMBOL FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51
Q36. REMOVE NEED FOR ALL SYMBOLS FOR 1X50 AS INDICATED ON 1X51.
Q13. DELETE SUPERPHRASE FROM IM, MOST RECENT OCCURRENCE ONLY.
Q22. DO SYNTAX OF CURRENT.
Q23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
CURRENT WORD IS...TO
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =DECIDED  ,ATT =F6,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E21
    WORD  =DECIDED  ,ATT =F6,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
Q23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
CURRENT WORD IS...TABLE
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =TO  ,ATT =F1,CURRENT TO BE SYMB
    *POSSIBLE.ASSIGN -TO- TO -TABLE- VIA -F1
Q07. MAKE 1X50 A VALUE OF 1X60 ON 1X70.
```

```
Q12. REMOVE NEED OF SYMBOL FOR 1X60 ON 1X50.
Q06. REMOVE NEED OF SYMBOL FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q36. REMOVE NEED FOR ALL SYMBOLS FOR 1X50 AS INDICATED ON 1X51.
Q13. DELETE SUPERPHRASE FROM IM, MOST RECENT OCCURRENCE ONLY.
Q22. DO SYNTAX OF CURRENT.
    WORD  =DECIDED  ,ATT =F6,CURRENT TO BE VALU
    *POSSIBLE.ASSIGN -TABLE- TO -DECIDED- VIA -F6
Q08. MAKE 1X70 A VALUE OF 1X60 ON 1X50.
Q10. REMOVE NEED OF VALUE FOR 1X60 ON 1X50.
Q05. REMOVE NEED OF VALUE FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q38. REMOVE ALL NEEDS FOR SYMBOL BY CURRENT.
Q34. DO SYNTAX OF SUPERPHRASE.
CURRENT WORD IS...THE
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =TABLE  ,ATT =F59,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =TABLE  ,ATT =F8,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =TABLE  ,ATT =F10,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =DECIDED  ,ATT =F6,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E21
    WORD  =TABLE  ,ATT =F59,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =TABLE  ,ATT =F8,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =TABLE  ,ATT =F10,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =DECIDED  ,ATT =F6,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
Q23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
CURRENT WORD IS...BILL
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =THE  ,ATT =F70,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =THE  ,ATT =F,CURRENT TO BE SYMB
    *POSSIBLE.ASSIGN -THE- TO -BILL- VIA -F
Q07. MAKE 1X50 A VALUE OF 1X60 ON 1X70.
Q12. REMOVE NEED OF SYMBOL FOR 1X60 ON 1X50.
Q06. REMOVE NEED OF SYMBOL FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q36. REMOVE NEED FOR ALL SYMBOLS FOR 1X50 AS INDICATED ON 1X51.
Q13. DELETE SUPERPHRASE FROM IM, MOST RECENT OCCURRENCE ONLY.
Q22. DO SYNTAX OF CURRENT.
    WORD  =TABLE  ,ATT =F59,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =TABLE  ,ATT =F8,CURRENT TO BE VALU
    *POSSIBLE.ASSIGN -BILL- TO -TABLE- VIA -F8
Q08. MAKE 1X70 A VALUE OF 1X60 ON 1X50.
Q10. REMOVE NEED OF VALUE FOR 1X60 ON 1X50.
Q05. REMOVE NEED OF VALUE FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q38. REMOVE ALL NEEDS FOR SYMBOL BY CURRENT.
```

```
 Q34. DO SYNTAX OF SUPERPHRASE.
 CURRENT WORD IS...UNTIL
 Q22. DO SYNTAX OF CURRENT.
 PROCESS CURRENT PHRASE.
   GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
      WORD  =TABLE   ,ATT =F59,CURRENT TO BE VALU
      IMPOSSIBLE. EXIT TO GENERATE NEXT.
      WORD  =TABLE   ,ATT =F10,CURRENT TO BE VALU
      *POSSIBLE.ASSIGN -UNTIL- TO -TABLE- VIA -F10
 Q08. MAKE 1X70 A VALUE OF 1X60 ON 1X50.
 Q10. REMOVE NEED OF VALUE FOR 1X60 ON 1X50.
 Q05. REMOVE NEED OF VALUE FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X
 Q38. REMOVE ALL NEEDS FOR SYMBOL BY CURRENT.
 Q34. DO SYNTAX OF SUPERPHRASE.
 CURRENT WORD IS...FURTHER
 Q22. DO SYNTAX OF CURRENT.
 PROCESS CURRENT PHRASE.
   GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
      WORD  =UNTIL   ,ATT =F12,CURRENT TO BE VALU
      IMPOSSIBLE. EXIT TO GENERATE NEXT.
      WORD  =TABLE   ,ATT =F59,CURRENT TO BE VALU.
      IMPOSSIBLE. EXIT TO GENERATE NEXT.
      WORD  =DECIDED  ,ATT =F6,CURRENT TO BE VALU
      IMPOSSIBLE. EXIT TO GENERATE NEXT.
   GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E21
      WORD  =UNTIL   ,ATT =F12,CURRENT TO BE VALU
      IMPOSSIBLE. EXIT TO GENERATE NEXT.
      WORD  =TABLE   ,ATT =F59,CURRENT TO BE VALU
      IMPOSSIBLE. EXIT TO GENERATE NEXT.
      WORD  =DECIDED  ,ATT =F6,CURRENT TO BE VALU
      IMPOSSIBLE. EXIT TO GENERATE NEXT.
 Q23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
 CURRENT WORD IS...NOTICE
 Q22. DO SYNTAX OF CURRENT.
 PROCESS CURRENT PHRASE.
   GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
      WORD  =FURTHER  ,ATT =F14,CURRENT TO BE SYMB
      *POSSIBLE.ASSIGN -FURTHER- TO -NOTICE- VIA -F14
 Q07. MAKE 1X50 A VALUE OF 1X60 ON 1X70.
 Q12. REMOVE NEED OF SYMBOL FOR 1X60 ON 1X50.
 Q06. REMOVE NEED OF SYMBOL FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X
 Q36. REMOVE NEED FOR ALL SYMBOLS FOR 1X50 AS INDICATED ON 1X51.
 Q13. DELETE SUPERPHRASE FROM IM, MOST RECENT OCCURRENCE ONLY.
 Q22. DO SYNTAX OF CURRENT.
      WORD  =UNTIL   ,ATT =F12,CURRENT TO BE VALU
      *POSSIBLE.ASSIGN -NOTICE- TO -UNTIL- VIA -F12
 Q08. MAKE 1X70 A VALUE OF 1X60 ON 1X50.
 Q10. REMOVE NEED OF VALUE FOR 1X60 ON 1X50.
 Q05. REMOVE NEED OF VALUE FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X5
 Q38. REMOVE ALL NEEDS FOR SYMBOL BY CURRENT.
 Q34. DO SYNTAX OF SUPERPHRASE.
 END OF SENTENCE.....
 Q23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
                                  *****
                                  X0001
                                  *****
```

```
DECIDED( F4--COMMITTEE( F--THE()),F6--TABLE( F1--TO(),F8--
BILL( F--THE()),F10--UNTIL( F12--NOTICE( F14--FURTHER())))))


                    *****
                    X0070
                    *****
 ***()
```

```
URRENT SENTENCE IS....
HE     NOTICE     YOU     WERE     IN     SEARCH     OF     IS     ON     THE
     TABLE      .
URRENT WORD IS...THE
22. DO SYNTAX OF CURRENT.
ROCESS CURRENT PHRASE.
 GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
 GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E21
23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
URRENT WORD IS...NOTICE
22. DO SYNTAX OF CURRENT.
ROCESS CURRENT PHRASE.
 GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =THE   ,ATT =F70,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =THE   ,ATT =F,CURRENT TO BE SYMB
    *POSSIBLE.ASSIGN -THE- TO -NOTICE- VIA -F
07. MAKE 1X50 A VALUE OF 1X60 ON 1X70.
12. REMOVE NEED OF SYMBOL FOR 1X60 ON 1X50.
06. REMOVE NEED OF SYMBOL FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
36. REMOVE NEED FOR ALL SYMBOLS FOR 1X50 AS INDICATED ON 1X51.
13. DELETE SUPERPHRASE FROM IM, MOST RECENT OCCURRENCE ONLY.
22. DO SYNTAX OF CURRENT.
23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
URRENT WORD IS...YOU
22. DO SYNTAX OF CURRENT.
ROCESS CURRENT PHRASE.
 GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =NOTICE   ,ATT =F3,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE   ,ATT =F16,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE   ,ATT =F36,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
 GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E21
    WORD  =NOTICE   ,ATT =F3,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE   ,ATT =F16,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE   ,ATT =F36,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
URRENT WORD IS...WERE
22. DO SYNTAX OF CURRENT.
ROCESS CURRENT PHRASE.
 GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =YOU   ,ATT =F25,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =YOU   ,ATT =F24,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =YOU   ,ATT =F33,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =YOU   ,ATT =F41,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =YOU   ,ATT =F34,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =YOU  ,ATT =F3,CURRENT TO BE SYMB
    *POSSIBLE.ASSIGN -YOU- TO -WERE- VIA -F3
Q07. MAKE 1X50 A VALUE OF 1X60 ON 1X70.
Q12. REMOVE NEED OF SYMBOL FOR 1X60 ON 1X50.
Q06. REMOVE NEED OF SYMBOL FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q36. REMOVE NEED FOR ALL SYMBOLS FOR 1X50 AS INDICATED ON 1X51.
Q13. DELETE SUPERPHRASE FROM IM, MOST RECENT OCCURRENCE ONLY.
Q22. DO SYNTAX OF CURRENT.
    WORD  =NOTICE  ,ATT =F3,CURRENT TO BE SYMB
    *POSSIBLE.ASSIGN -NOTICE- TO -WERE- VIA -F3
Q07. MAKE 1X50 A VALUE OF 1X60 ON 1X70.
Q12. REMOVE NEED OF SYMBOL FOR 1X60 ON 1X50.
Q06. REMOVE NEED OF SYMBOL FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q36. REMOVE NEED FOR ALL SYMBOLS FOR 1X50 AS INDICATED ON 1X51.
Q13. DELETE SUPERPHRASE FROM IM, MOST RECENT OCCURRENCE ONLY.
Q22. DO SYNTAX OF CURRENT.
    WORD  =NOTICE  ,ATT =F16,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE  ,ATT =F36,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
Q23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
CURRENT WORD IS...IN
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =WERE  ,ATT =F5,CURRENT TO BE VALU
    *POSSIBLE.ASSIGN -IN- TO -WERE- VIA -F5
Q08. MAKE 1X70 A VALUE OF 1X60 ON 1X50.
Q10. REMOVE NEED OF VALUE FOR 1X60 ON 1X50.
Q05. REMOVE NEED OF VALUE FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q38. REMOVE ALL NEEDS FOR SYMBOL BY CURRENT.
Q34. DO SYNTAX OF SUPERPHRASE.
CURRENT WORD IS...SEARCH
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =IN  ,ATT =F18,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IN  ,ATT =F55,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IN  ,ATT =F15,CURRENT TO BE VALU
    *POSSIBLE.ASSIGN -SEARCH- TO -IN- VIA -F15
Q08. MAKE 1X70 A VALUE OF 1X60 ON 1X50.
Q10. REMOVE NEED OF VALUE FOR 1X60 ON 1X50.
Q05. REMOVE NEED OF VALUE FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q38. REMOVE ALL NEEDS FOR SYMBOL BY CURRENT.
Q34. DO SYNTAX OF SUPERPHRASE.
CURRENT WORD IS...OF
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =SEARCH  ,ATT =F56,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =SEARCH  ,ATT =F19,CURRENT TO BE VALU
    *POSSIBLE.ASSIGN -OF- TO -SEARCH- VIA -F19
Q08. MAKE 1X70 A VALUE OF 1X60 ON 1X50.
```

```
Q10. REMOVE NEED OF VALUE FOR 1X60 ON 1X50.
Q05. REMOVE NEED OF VALUE FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q38. REMOVE ALL NEEDS FOR SYMBOL BY CURRENT.
Q34. DO SYNTAX OF SUPERPHRASE.
CURRENT WORD IS...IS
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =SEARCH   ,ATT =F56,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IN  ,ATT =F18,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IN  ,ATT =F55,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE  ,ATT =F16,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE  ,ATT =F36,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =WERE  ,ATT =F5,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E21
    WORD  =SEARCH   ,ATT =F56,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IN  ,ATT =F18,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IN  ,ATT =F55,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE  ,ATT =F16,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE  ,ATT =F36,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =WERE  ,ATT =F5,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
Q23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
CURRENT WORD IS...ON
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =IS  ,ATT =F54,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IS  ,ATT =F3,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IS  ,ATT =F5,CURRENT TO BE VALU
    *POSSIBLE.ASSIGN -ON- TO -IS- VIA -F5
Q08. MAKE 1X70 A VALUE OF 1X60 ON 1X50.
Q10. REMOVE NEED OF VALUE FOR 1X60 ON 1X50.
Q05. REMOVE NEED OF VALUE FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q38. REMOVE ALL NEEDS FOR SYMBOL BY CURRENT.
Q34. DO SYNTAX OF SUPERPHRASE.
CURRENT WORD IS...THE
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =IS  ,ATT =F54,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =ON  ,ATT =F48,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =ON   ,ATT =F20,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IS   ,ATT =F3,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IS   ,ATT =F67,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IS   ,ATT =F5,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =SEARCH   ,ATT =F56,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IN   ,ATT =F18,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IN   ,ATT =F55,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE   ,ATT =F16,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE   ,ATT =F36,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =WERE   ,ATT =F5,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E21
    WORD  =IS   ,ATT =F54,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =ON   ,ATT =F48,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =ON   ,ATT =F20,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IS   ,ATT =F3,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IS   ,ATT =F67,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IS   ,ATT =F5,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =SEARCH   ,ATT =F56,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IN   ,ATT =F18,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =IN   ,ATT =F55,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE   ,ATT =F16,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =NOTICE   ,ATT =F36,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =WERE   ,ATT =F5,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
Q23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
CURRENT WORD IS...TABLE
Q22. DO SYNTAX OF CURRENT.
PROCESS CURRENT PHRASE.
  GENERATE THE FOLLOWING PROPOSALS FOR SUBPROCESS E24
    WORD  =THE   ,ATT =F70,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =THE   ,ATT =F,CURRENT TO BE SYMB
    *POSSIBLE.ASSIGN -THE- TO -TABLE- VIA -F
Q07. MAKE 1X50 A VALUE OF 1X60 ON 1X70.
```

```
Q12. REMOVE NEED OF SYMBOL FOR 1X60 ON 1X50.
Q06. REMOVE NEED OF SYMBOL FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q36. REMOVE NEED FOR ALL SYMBOLS FOR 1X50 AS INDICATED ON 1X51.
Q13. DELETE SUPERPHRASE FROM IM, MOST RECENT OCCURRENCE ONLY.
Q22. DO SYNTAX OF CURRENT.
    WORD  =IS  ,ATT =F54,CURRENT TO BE SYMB
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =ON  ,ATT =F48,CURRENT TO BE VALU
    IMPOSSIBLE. EXIT TO GENERATE NEXT.
    WORD  =ON  ,ATT =F20,CURRENT TO BE VALU
    *POSSIBLE.ASSIGN -TABLE- TO -ON- VIA -F20
Q08. MAKE 1X70 A VALUE OF 1X60 ON 1X50.
Q10. REMOVE NEED OF VALUE FOR 1X60 ON 1X50.
Q05. REMOVE NEED OF VALUE FOR 1X60 BY 1X50 AS INDICATED ON SUPERPHRASE 1X51.
Q38. REMOVE ALL NEEDS FOR SYMBOL BY CURRENT.
Q34. DO SYNTAX OF SUPERPHRASE.
END OF SENTENCE.....
Q23. ADD CURRENT PHRASE AS MOST RECENT IM ENTRY.
                                  *****
                                  X0001
                                  *****
                  WERE( F3--YOU()--NOTICE( F--THE()),F5--IN( F15--SEARCH( F19--
                  OF()))


                                  *****
                                  X0002
                                  *****
                  IS( F5--ON( F20--TABLE( F--THE()))))


                                  *****
                                  X0070
                                  *****
                   ***()
```

processed by JIGSAW-1 while sentence 1 was not. It is instructive to analyze the failure in the second example.

Two disjoint pieces resulted because "notice" could not be assigned to "is" since this would violate the tree constraint. This is a direct result of the incorrect assignment of "notice" as a dependent of "were" rather than the other way around. The assignment actually made was proposed first since "were" occurs later in the sentence and current words are used as governors before they are used as dependents. The assignment made uses label F3, which reflects a sense in which "notice" acts as a subject of the verb "were", as in "The notice were late". This particular difficulty could be eliminated by providing a larger selection of labels so that subjects and verbs are forced to agree in number. However, this would defeat the purpose of the over-general linguistic description. Actually, the corpus from which the program was initialized did not contain the "ungrammatical" construction "notice were". It did, however, contain a sentence in which "notice" is a subject of a copulative verb and a sentence in which "were" is used as a copulative verb, but with another subject. Since the dictionary contains no information about particular word pairs, the assignment of "notice" to "were" is not rejected by the dictionary.

If such situations are not to be ruled out by the dictionary, how can such clumsy failures be avoided? The answer lies in the syntax tree. Somehow, the program should know that "were" is no longer acceptable as the verb for "notice" since the former already has a subject, namely "you". Furthermore, no conjunction exists to tie "notice" and "you" into a compound subject for "were". But unless the syntax tree contains tests which detect the substructure of the "were"-phrase, these facts will go undetected and the resulting error is perfectly natural. This example indicates the need for subphrase syntax, not present in JIGSAW-1. The problem is further discussed in the ensuing experimental report.

It should be added, however, that the sentences employed as examples are fairly sophisticated examples of English. JIGSAW-1's aspiration level is such as to be undaunted by this failure. A program without subphrase syntax will surely be able to handle simple sentences of the sort used and understood by a child of three or four. Clearly, the program must have the ability to learn subphrase syntax if it is to progress to adult linguistic abilities. But such progress, if possible within JIGSAW's framework, will undoubtedly require a careful and elaborate education process of an order of magnitude comparable to that employed by humans. To expect otherwise is perhaps asking too much of a science which is only beginning to learn how to babble.

AN EXPERIMENT WITH JIGSAW-1

In order to explore the feasibility of the approach outlined in the previous sections, an experiment was performed on a small sample of English sentences. The 25 sentences used were composed and diagrammed by assistants unfamiliar with the operation of JIGSAW-1. The assistants were instructed to prepare sentences of reasonable length such that the total vocabulary employed was less than 50 distinct words. The last condition was imposed to encourage the use of some words in a variety of syntactic and semantic contexts. The corpus produced was in no way edited by the programmer.

The sentences are presented in diagrammed form in the Appendix. The dictionary and syntax routines were derived by the algorithm presented in a previous section; this material also appears in the Appendix. It will be noted that the sentences employ some moderately complex constructions. It was not anticipated, and did not prove to be the case, that JIGSAW-1 was able to produce the sentence diagrams from which the linguistic description derived. The failures, however, point to the limitations of the method as presently constituted, and the successes give some indication of the extent to which word position cues alone can direct linguistic processing. The results as produced by the program appear on the next pages. Brief comments on each analysis follow.

Sentence 0	Processed correctly.
Sentence 1	Error as noted in the previous section. This difficulty may be corrected by the addition of subphrase syntax (SPS).
Sentence 2	Processed correctly.
Sentence 3	The program failed to handle the compound predicate; this is typical behavior. JIGSAW-1 does not understand English conjunctions, and will assume (and form) a simple phrase without looking for conjunctions. This may be corrected by allowing tests on function words to be used in the syntax tree.
Sentence 4	Processed correctly.
Sentence 5	Processed correctly.
Sentence 6	"Table" was incorrectly connected to "decided" as an object rather than to "was" as a subject. This may be corrected either by SPS or a semantic check: One may decide to table, but one does not decide the table.
Sentence 7	"The committee" is incorrectly used as a second object of "for". This reflects the frequent error of allowing proposals produced at a remote (early) point in the sentence to take precedence over connections which would be proposed for the current word and could be filled by words immediately following. This may be corrected by SPS which removes some proposals once some action is taken.

Sentence 8	Processed correctly.
Sentence 9	Processed correctly.
Sentence 10	Processed correctly.
Sentence 11	Again a compound predicate was handled incorrectly as in Sentence 3. Also "spend"-"little" was selected instead of "little"-"time" and "and"-"money" instead of "buy"-"money"; this may be remedied by SPS to remove the influence of remote connections, as noted in connection with Sentence 7.
Sentence 12	Again, failure to handle compound predicates.
Sentence 13	Failure for reason given with Sentence 7.
Sentence 14	Processed correctly.
Sentence 15	"More"-"work" connection bears the wrong label. This problem must be solved by a more sophisticated semantic dictionary or more judicious selection of labels.
Sentence 16	"In"-"much" was selected instead of "in"-"make". This is also a problem of remote proposals dominating; see Sentence 7.
Sentence 17	The failure to produce a single structure is due to the incorrect structure "much"-"do" (instead of "much"-"work". This problem is more serious and must either be solved by more judicious selection of labels in the corpus diagrams, SPS of a fairly complex sort, or a revision in the basic JIGSAW-1 order of examining proposals.

Sentence 18	Processed correctly.
Sentence 19	The failure here is traceable to the difficulty experienced with compound predicates.
Sentence 20	Both instances of "it" have been combined with the same instance of "is". This may be remedied with SPS to eliminate remote connections. The reversal of the "is"-"and" construction is a more basic fault deriving from JIGSAW-1's strategy (in E24) of using the current symbol as a governor before attempting to use it as a dependent.
Sentence 21	Processed correctly.
Sentence 22	The failure here is for the same reasons outlined for Sentence 17.
Sentence 23	"It" was combined as the object of the first instance of "buy" rather than as the subject of the second instance of "buy". This may be corrected with SPS to suppress remote proposals.
Sentence 24	SPS to suppress remote proposals would also eliminate two errors in the processing of this sentence: the incorrect "little"-"spend" and "time"-"in". Also the difficulty with compound predicates again appears.

In summary, 10 of 25 sentences were processed correctly; of the remainder, all but two appear to be completely within the range of JIGSAW with an extension of the syntax tree to allow tests which deal with function words and the nature of the substructure of the phrase whose syntax routine is being recovered.

It remains to be determined (a) how the SPS capability is to be implemented, and (b) if JIGSAW can proceed with its learning strategy from partial information about a corpus to a state where it can handle the entire corpus.

```
CURRENT SENTENCE IS....
THE      COMMITTEE      DECIDED      TO      TABLE      THE      BILL      UNTIL
FURTHER      NOTICE      .
                                   *****
                                   X0001
                                   *****
                DECIDED( F4--COMMITTEE( F--THE()),F6--TABLE( F1--TO(),F8--
                BILL( F--THE()),F10--UNTIL( F12--NOTICE( F14--FURTHER())))))


                                   *****
                                   X0070
                                   *****
                 ***()


CURRENT SENTENCE IS....
THE      NOTICE      YOU      WERE      IN      SEARCH      OF      IS      ON      THE
     TABLE      .
                                   *****
                                   X0001
                                   *****
            ,   WERE( F3--YOU()--NOTICE( F--THE()),F5--IN( F15--SEARCH( F19--
                OF()))


                                   *****
                                   X0002
                                   *****
                IS( F5--ON( F20--TABLE( F--THE())))


                                   *****
                                   X0070
                                   *****
                 ***()


CURRENT SENTENCE IS....
BILL      THOUGHT      THE      COMMITTEE      DECIDED      TO      SPEND      MORE
     TIME      ON      LESS      REWARDING      WORK      .
                                   *****
                                   X0001
                                   *****
                THOUGHT( F43--BILL(),F45--DECIDED( F4--COMMITTEE( F--THE()),
                F6--SPEND( F1--TO(),F11--TIME( F7--MORE(),F61--ON( F48--WORK(
                F47--REWARDING( F7--LESS()))))))))


                                   *****
                                   X0070
                                   *****
                 ***()
```

```
CURRENT SENTENCE IS....
THE      COMMITTEE      DECIDED       NOT       TO       SPEND       MORE       MONEY
AND      TO      TABLE       THE       BILL       FOR       A       LITTLE       MORE
TIME      .
                                        *****
                                        X0001
                                        *****
                      DECIDED( F4--COMMITTEE( F--THE()),F6--SPEND( F1--TO(),F49--
                      NOT(),F11--MONEY( F7--MORE()))--AND( F54--TABLE( F1--TO(),F8
                      --BILL( F--THE()),F59--FOR( F60--TIME( F7--MORE( F68--LITTLE(
                      F69--A()))))))))


                                        *****
                                        X0070
                                        *****
                       ***()


CURRENT SENTENCE IS....
BILL      WILL      MAKE       THE       TABLE       NOT       BUY       IT       .
                                        *****
                                        X0001
                                        *****
                      MAKE( F2--WILL(),F33--BILL(),F40--TABLE( F--THE()),F62--BUY(
                      F49--NOT(),F44--IT()))


                                        *****
                                        X0070
                                        *****
                       ***()


CURRENT SENTENCE IS....
BILL      AND      YOU      WILL      DO      THE      HARD      WORK      FOR      THE
     COMMITTEE      .
                                        *****
                                        X0001
                                        *****
                      DO( F2--WILL(),F24--AND( F9--BILL()--YOU()),F26--WORK( F28--
                      HARD( F--THE()),F30--FOR( F32--COMMITTEE( F--THE())))))


                                        *****
                                        X0070
                                        *****
                       ***()
```

```
CURRENT SENTENCE IS....
AFTER     FURTHER     SEARCH     THE     COMMITTEE     DECIDED     THE
TABLE     WAS     WORTH     MORE     MONEY     THAN     BILL     THOUGHT     IT
     WAS     WORTH     .
                                   *****
                                   X0001
                                   *****
                    DECIDED( F4--COMMITTEE( F--THE()),F50--AFTER( F51--SEARCH(
                    F14--FURTHER())),F6--TABLE( F--THE(),F8--BILL())--WAS( F5--
                    WORTH( F52--MONEY( F7--MORE( F21--THAN( F22--THOUGHT( F45--
                    WAS( F5--WORTH()))))))),F3--IT()))


                                   *****
                                   X0070
                                   *****
                     ***()


CURRENT SENTENCE IS....
THE     SEARCH     FOR     A     TABLE     WAS     A     SUCCESS     AND
THE     COMMITTEE     DECIDED     TO     BUY     IT     .
                                   *****
                                   X0001
                                   *****
                    AND( F54--WAS( F3--SEARCH( F--THE(),F19--FOR( F65--TABLE( F27
                    --A()),F32--COMMITTEE( F--THE()))))--SUCCESS( F27--A()))--
                    DECIDED( F6--BUY( F1--TO(),F44--IT()))))


                                   *****
                                   X0070
                                   *****
                     ***()


CURRENT SENTENCE IS....
YOU     WILL     NOTICE     THAT     BILL     WILL     WORK     HARD     FOR
     LITTLE     MONEY     .
                                   *****
                                   X0001
                                   *****
                    NOTICE( F2--WILL(),F34--YOU(),F36--THAT( F37--WORK( F2--WILL(
                    ),F41--BILL(),F28--HARD(),F30--FOR( F39--MONEY( F7--LITTLE())
                    ))))


                                   *****
                                   X0070
                                   *****
                     ***()
```

```
CURRENT SENTENCE IS....
AFTER     FURTHER     THOUGHT     BILL     DECIDED     TO     MAKE     THE
SEARCH     COMMITTEE     WORK     .
                                  *****
                                  X0001
                                  *****
                  DECIDED( F4--BILL(),F50--AFTER( F51--THOUGHT( F14--FURTHER())
                  ),F6--MAKE( F1--TO(),F40--SEARCH( F--THE(),F56--WORK( F72--
                  COMMITTEE())))))


                                  *****
                                  X0070
                                  *****
                   ***()


CURRENT SENTENCE IS....
TO     BUY     MUCH     AND     SPEND     LITTLE     IS     HARD     TO     DO
   .
                                  *****
                                  X0001
                                  *****
                  IS( F3--AND( F57--BUY( F1--TO(),F44--MUCH())--SPEND( F11--
                  LITTLE())),F5--HARD( F58--DO( F1--TO())))


                                  *****
                                  X0070
                                  *****
                   ***()


CURRENT SENTENCE IS....
YOU     CAN     SPEND     A     HARD     LIFE     AND     MAKE     MUCH
MONEY     BUT     MONEY     WILL     BUY     LITTLE     TIME     .
                                  *****
                                  X0001
                                  *****
                  BUT( F46--AND( F54--SPEND( F35--CAN(),F25--YOU(),F11--LIFE(
                  F28--HARD( F69--A()))--LITTLE())--MAKE( F40--MONEY( F7--MUCH(
                  ))),F9--MONEY()),F71--BUY( F2--WILL(),F44--TIME()))


                                  *****
                                  X0070
                                  *****
                   ***()
```

```
CURRENT SENTENCE IS....
YOU      CAN      MAKE      MONEY      IF      YOU      SPEND      MORE      TIME
AND      THOUGHT      AND      DO      HARD      WORK      .
                                    *****
                                    X0001
                                    *****
                   IF( F29--MAKE( F35--CAN(),F33--YOU(),F40--MONEY()),F31--SPEND
                   ( F25--YOU(),F11--TIME( F7--MORE(),F13--AND( F9--THOUGHT())))
                   --AND( F54--DO( F26--WORK( F28--HARD())))))


                                    *****
                                    X0070
                                    *****
                    ***()

CURRENT SENTENCE IS....
THE      MORE      MONEY      YOU      MAKE      THE      MORE      MONEY      YOU
SPEND      .
                                    *****
                                    X0001
                                    *****
                   MAKE( F33--YOU(),F40--MONEY( F7--MORE( F70--THE()))--MONEY(
                   F7--MORE( F70--THE())),F67--SPEND( F25--YOU()))


                                    *****
                                    X0070
                                    *****
                    ***()

CURRENT SENTENCE IS....
TIME      IS      MONEY      SPEND      IT      WISELY      .
                                    *****
                                    X0001
                                    *****
                   IS( F3--TIME()--MONEY(),F67--SPEND( F11--IT(),F38--WISELY()))

                                    *****
                                    X0070
                                    *****
                    ***()
```

```
CURRENT SENTENCE IS....
THE      MORE      YOU      WORK      THE      MORE      MONEY      YOU      WILL
MAKE      .
                                        *****
                                        X0001
                                        *****
                     WORK( F41--YOU(),F7--MORE( F70--THE()),F67--MAKE( F2--WILL(),
                     F33--YOU(),F40--MONEY( F7--MORE( F70--THE())))))


                                        *****
                                        X0070
                                        *****
                      ***()


CURRENT SENTENCE IS....
IT      IS      HARD      TO      MAKE      MUCH      . MONEY      IN      A      LITTLE
     TIME      .
                                        *****
                                        X0001
                                        *****
                     IS( F3--IT(),F5--HARD( F58--MAKE( F1--TO(),F40--MONEY( F7--
                     MUCH( F53--IN( F55--TIME( F7--LITTLE( F69--A()))))))))


                                        *****
                                        X0070
                                        *****
                      ***()


CURRENT SENTENCE IS....
IF      YOU      DO      MUCH      WORK      YOU      WILL      SPEND      A      HARD
     LIFE      .
                                        *****
                                        X0001
                                        *****
                     IF( F29--DO( F24--YOU(),F26--MUCH()))


                                        *****
                                        X0002
                                        *****
                     WORK( F67--SPEND( F2--WILL(),F25--YOU(),F11--LIFE()),F28--
                     HARD( F69--A()))


                                        *****
                                        X0070
                                        *****
                      ***()
```

```
CURRENT SENTENCE IS....
YOU      CAN      MAKE      MONEY      WORK      FOR      YOU      IF      YOU
SPEND      IT      WISELY      .
                                        *****
                                        X0001
                                        *****
                     IF( F29--MAKE( F35--CAN(),F33--YOU(),F40--MONEY(),F66--WORK(
                     F30--FOR( F32--YOU()))),F31--SPEND( F25--YOU(),F11--IT(),F38
                     --WISELY()))


                                        *****
                                        X0070
                                        *****
                      ***()


CURRENT SENTENCE IS....
HAPPINESS      IS      MONEY      AND      THE      TIME      TO      SPEND      IT
      .
                                        *****
                                        X0001
                                        *****
                     AND( F54--IS( F3--HAPPINESS()--MONEY()),F9--TIME( F--THE(),
                     F13--SPEND( F1--TO(),F11--IT())))


                                        *****
                                        X0070
                                        *****
                      ***()


CURRENT SENTENCE IS....
IT      IS      HARD      TO      MAKE      MONEY      AND      IT      IS      HARD
      TO      SPEND      IT      WISELY      .
                                        *****
                                        X0001
                                        *****
                     IS( F3--AND( F54--IS( F3--IT()--IT(),F5--HARD( F58--MAKE( F1
                     --TO(),F40--MONEY()))))),F5--HARD( F58--SPEND( F1--TO(),F11--
                     IT(),F38--WISELY()))))


                                        *****
                                        X0070
                                        *****
                      ***()
```

```
CURRENT SENTENCE IS....
SUCCESS     IN      THE      SEARCH      FOR      HAPPINESS      IS      MUCH
MORE     REWARDING      THAN      SUCCESS      IN      THE      SEARCH      FOR
MONEY     .
                                        *****
                                        X0001
                                        *****
                     IS( F3--SUCCESS( F17--IN( F15--SEARCH( F--THE(),F19--FOR( F23
                     --HAPPINESS())))),F5--REWARDING( F7--MORE( F68--MUCH(),F21--
                     THAN( F22--SUCCESS( F17--IN( F15--SEARCH( F--THE(),F19--FOR(
                     F39--MONEY()))))))))


                                        *****
                                        X0070
                                        *****
                      ***()


CURRENT SENTENCE IS....
YOU      CAN      DO      MUCH      MORE      WORK      IN      LITTLE      TIME      IF
     YOU      SPEND      TIME      WISELY      .
                                        *****
                                        X0001
                                        *****
                     IF( F29--DO( F35--CAN(),F24--YOU(),F26--MUCH()--WORK( F7--
                     MORE(),F30--IN( F55--TIME( F7--LITTLE()))))),F31--SPEND( F25--
                     YOU(),F11--TIME(),F38--WISELY()))


                                        *****
                                        X0070
                                        *****
                      ***()


CURRENT SENTENCE IS....
MONEY      WILL      BUY      MUCH      BUT      IT      WILL      NOT      BUY
HAPPINESS      .
                                        *****
                                        X0001
                                        *****
                     BUT( F46--BUY( F2--WILL(),F42--MONEY(),F44--MUCH( F--IT()),F71
                     --BUY( F49--NOT(),F2--WILL(),F44--HAPPINESS()))


                                        *****
                                        X0070
                                        *****
                      ***()
```

```
CURRENT SENTENCE IS....
YOU      WILL     DO      MUCH      IN      LIFE      IF      YOU      WILL      SPEND
     A      LITTLE      MORE      TIME      AND      THOUGHT      .
                                        *****
                                        X0001
                                        *****
                    IF( F29--DO( F2--WILL(),F24--YOU(),F26--MUCH( F53--IN( F18--
                    LIFE(),F55--TIME( F7--MORE(),F13--AND( F9--THOUGHT()))))))),F31
                    --SPEND( F2--WILL(),F25--YOU(),F11--LITTLE( F69--A()))))


                                        *****
                                        X0070
                                        *****
                     ***()
```

APPENDIX

Materials used in the experiment

Sentence diagrams.

The following pages contain the labeled dependency trees from which JIGSAW-1 was initialized. The diagrams are meant to reflect the structure of the sentence; however, if the diagrams given differ from the reader's intuition in minor respects this does not invalidate the experiment.

Arrows are used to connect dependent words to their governors, with the arrow head near the governor. Labels appear to the right of their associated arrows; they take the form of underlined numbers. In the program, the labels are IPL symbols of the form FN where N is a one or two digit integer and corresponds to the numbers on the diagrams. The integer over the initial letter of each word designates the serial position of that word in the original sentence.

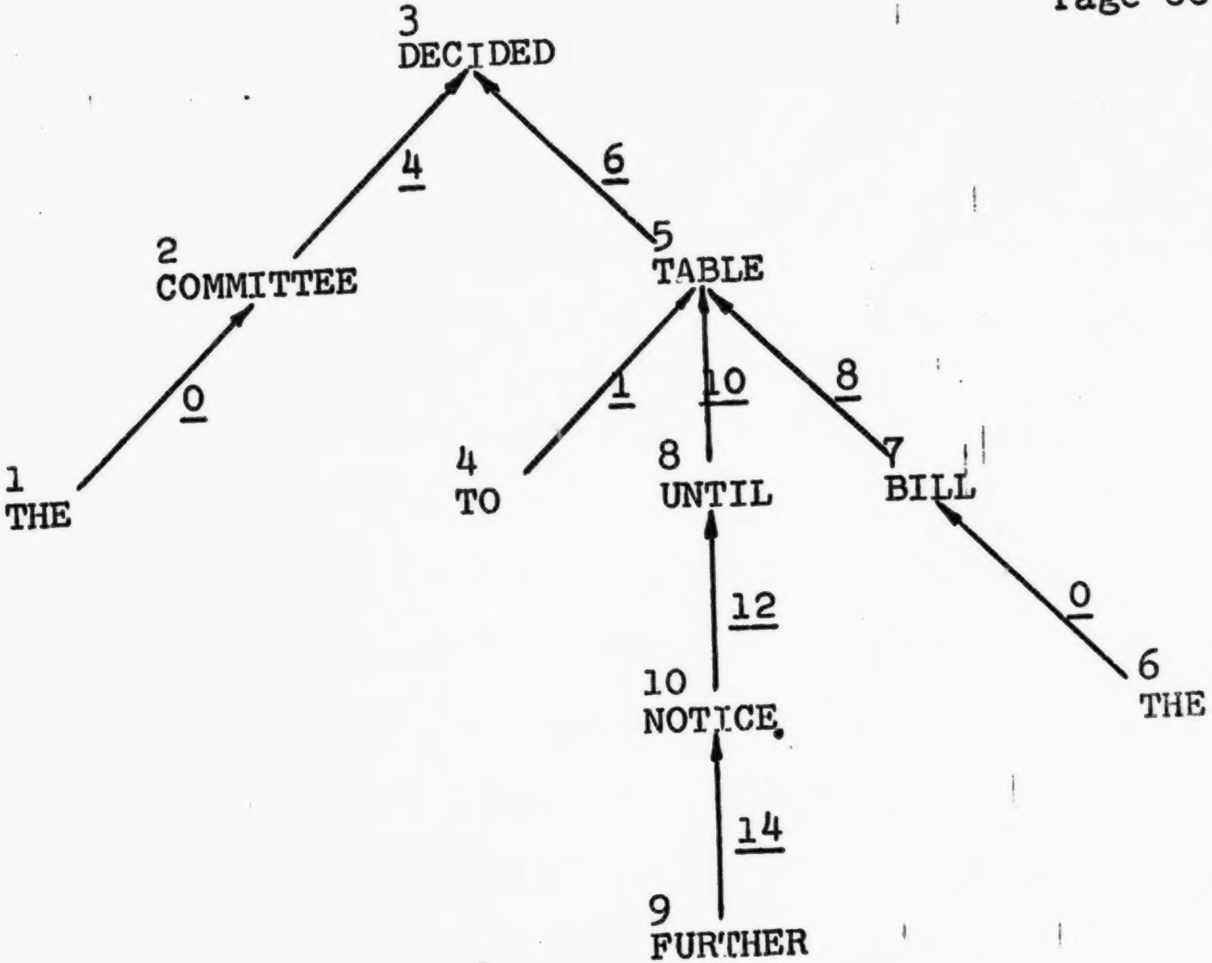
Labels.

An attempt has been made to assign a "sense" to each label. Naturally JIGSAW-1 is unaware of any such interpretation; to the program labels are merely arbitrary symbols. The table is included solely as an aid to the reader, and to provide an index to the occurrences of the labels.

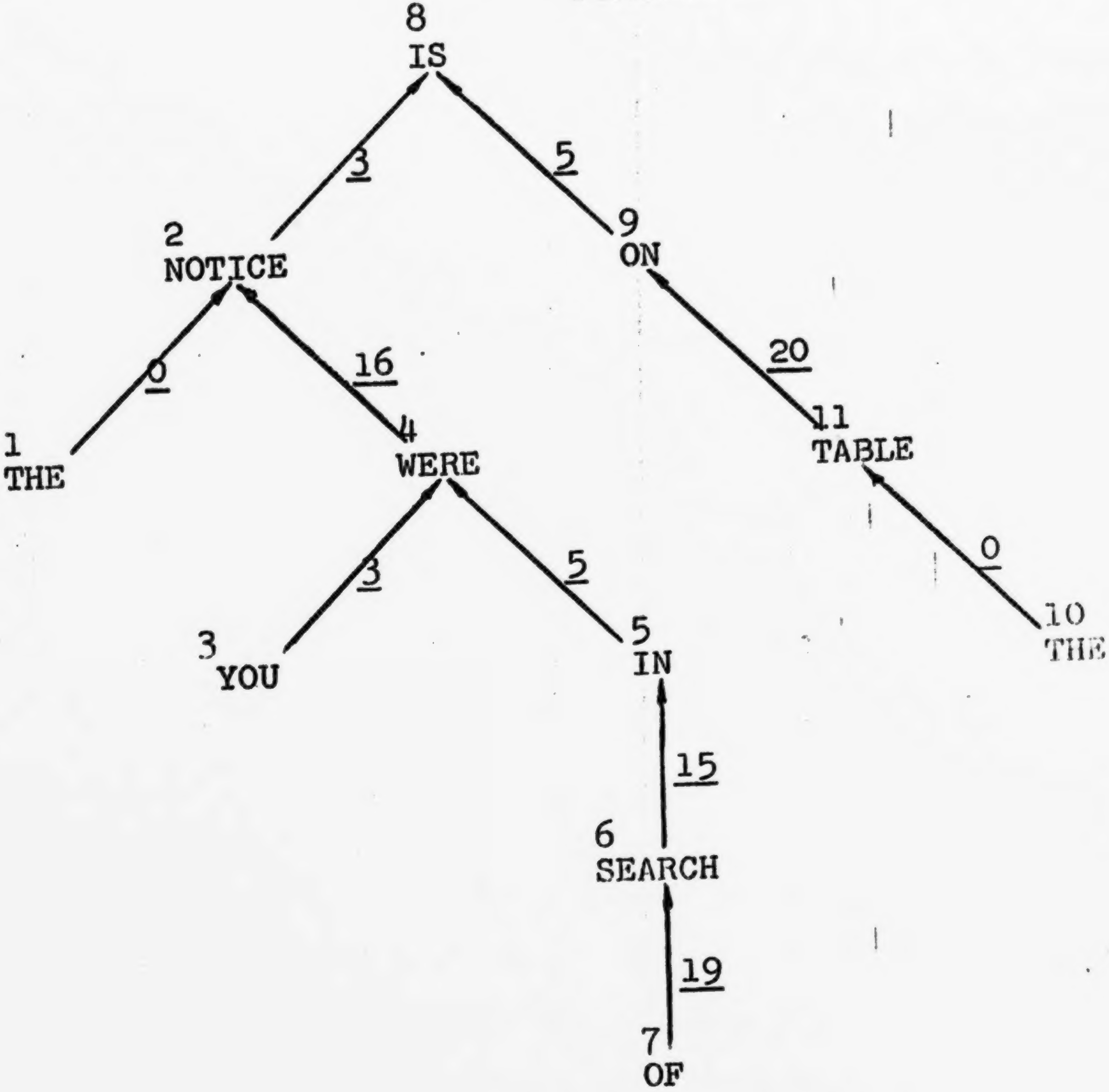
Syntax and semantics.

The information derived from the diagrams and used to form the dictionary and syntax routines is summarized in a table. For each word, the D3 list is composed of all the labels in the third column of the table; list D4 is composed of all the labels in the fourth column which are preceded by an asterisk; list Q101 is composed of all labels in the third column which are preceded by an asterisk; list Q100 is composed of all labels in the fourth column which are preceded by an asterisk.

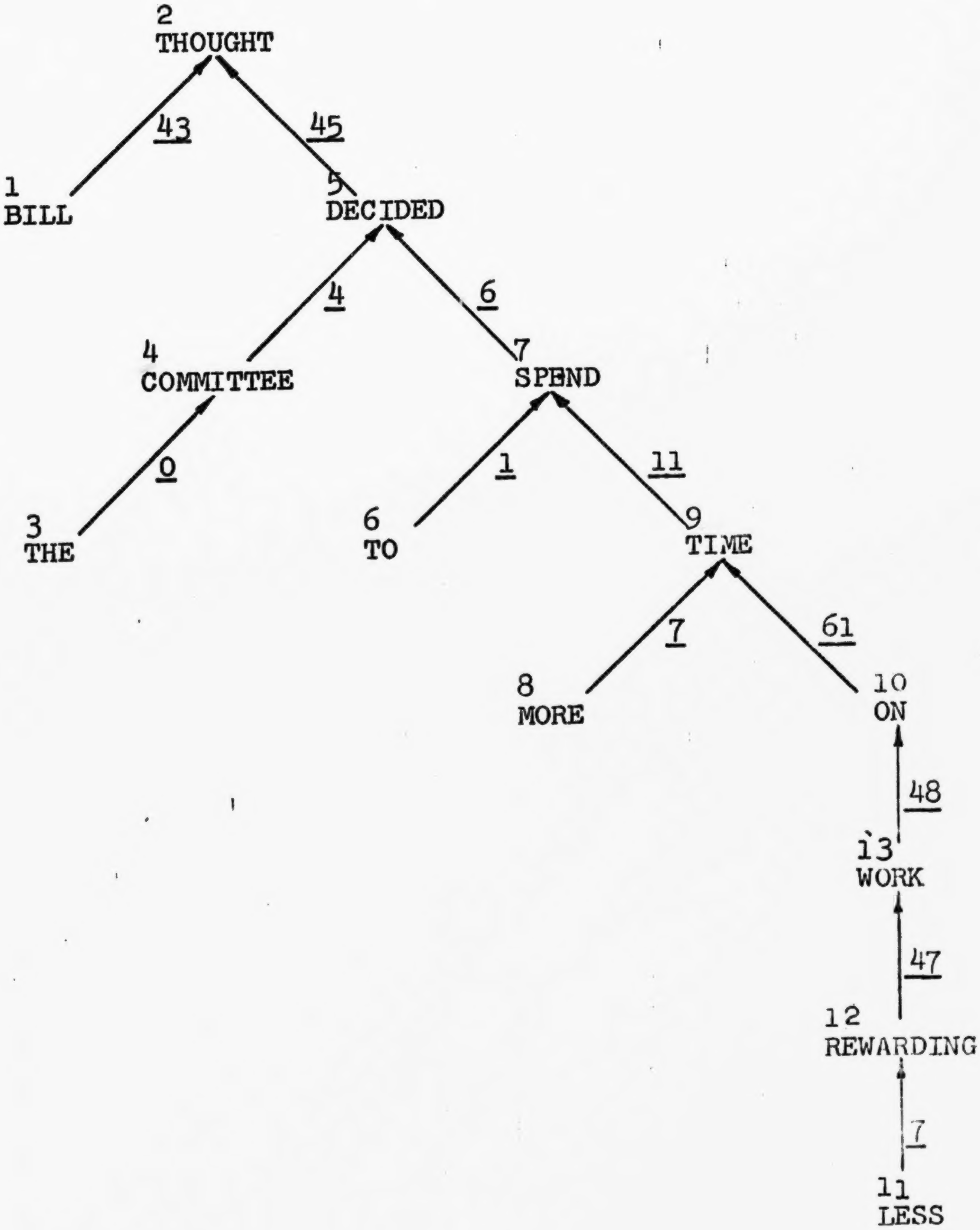
Sentence 0


3
DECIDED
4
6
2
COMMITTEE
5
TABLE
0
1
10
8
1
THE
4
TO
8
UNTIL
7
BILL
12
0
10
NOTICE
6
THE
14
9
FURTHER

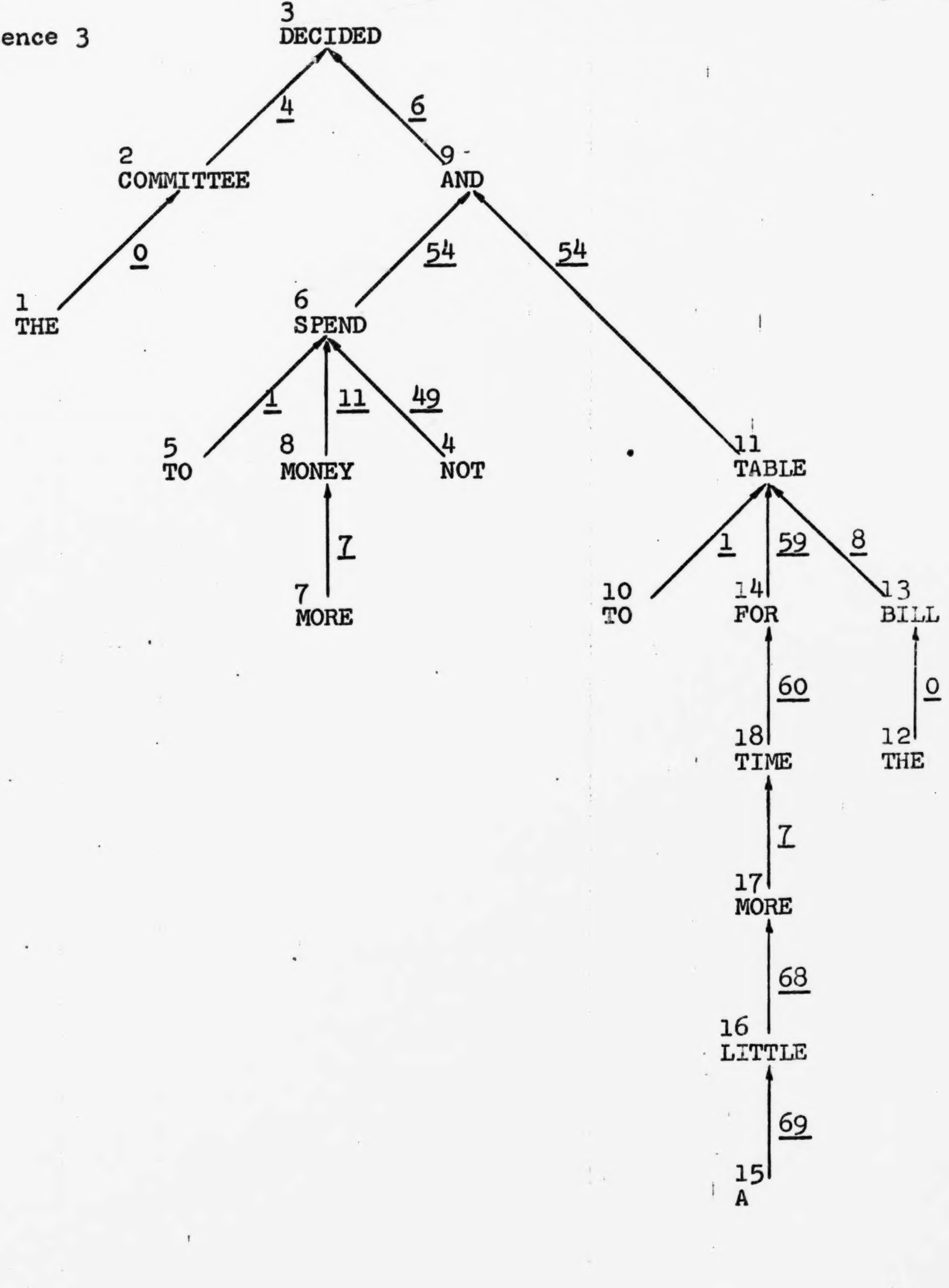
Sentence 1


8
IS
3
5
2
NOTICE
9
ON
0
16
20
1
THE
4
WERE
11
TABLE
3
5
0
3
YOU
5
IN
10
THE
15
6
SEARCH
19
7
OF

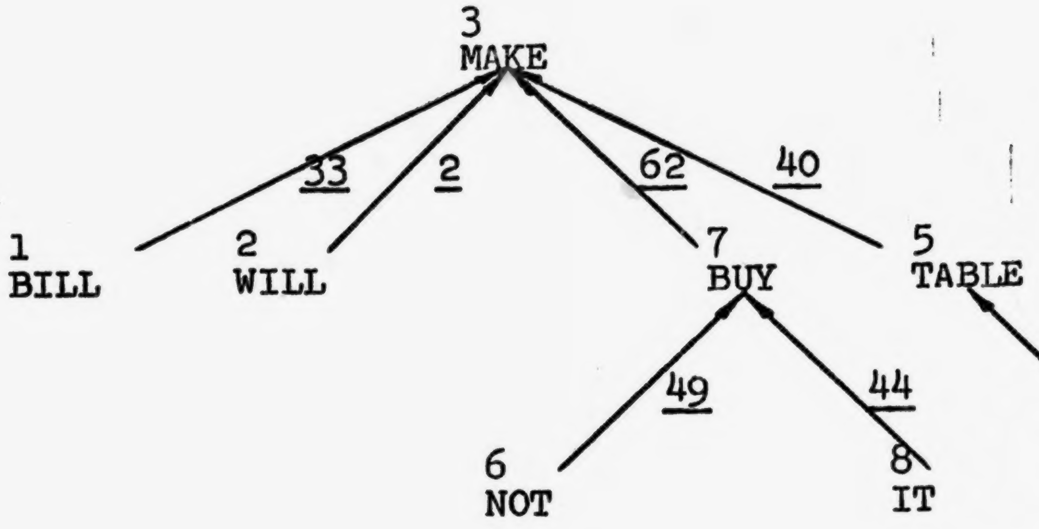
Sentence 2


THOUGHT
BILL
DECIDED
COMMITTEE
SPEND
THE
TO
TIME
MORE
ON
WORK
REWARDING
LESS

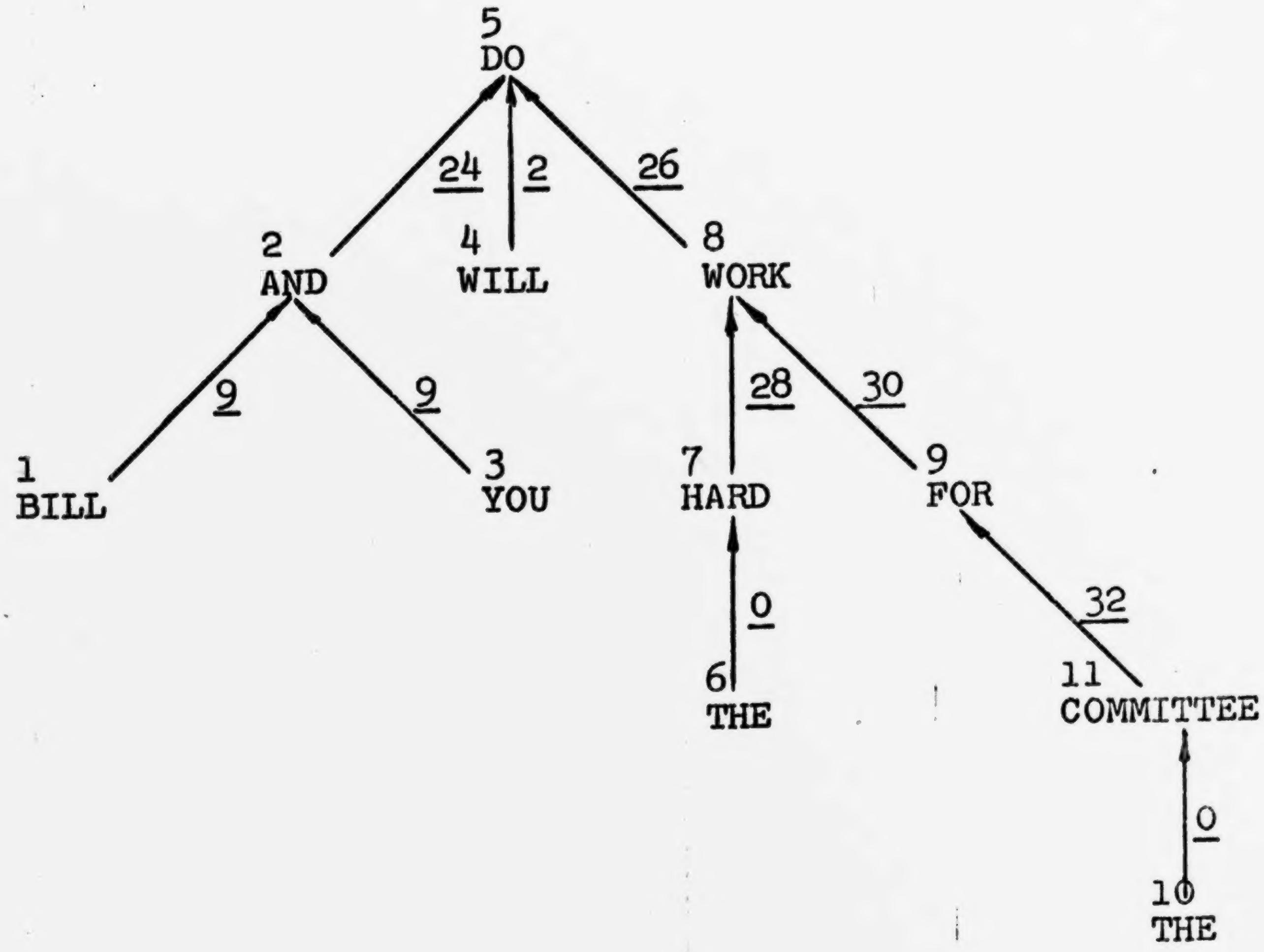
Sentence 3


DECIDED
COMMITTEE
AND
THE
SPEND
TO
MONEY
NOT
MORE
TABLE
TO
FOR
BILL
TIME
THE
MORE
LITTLE
A

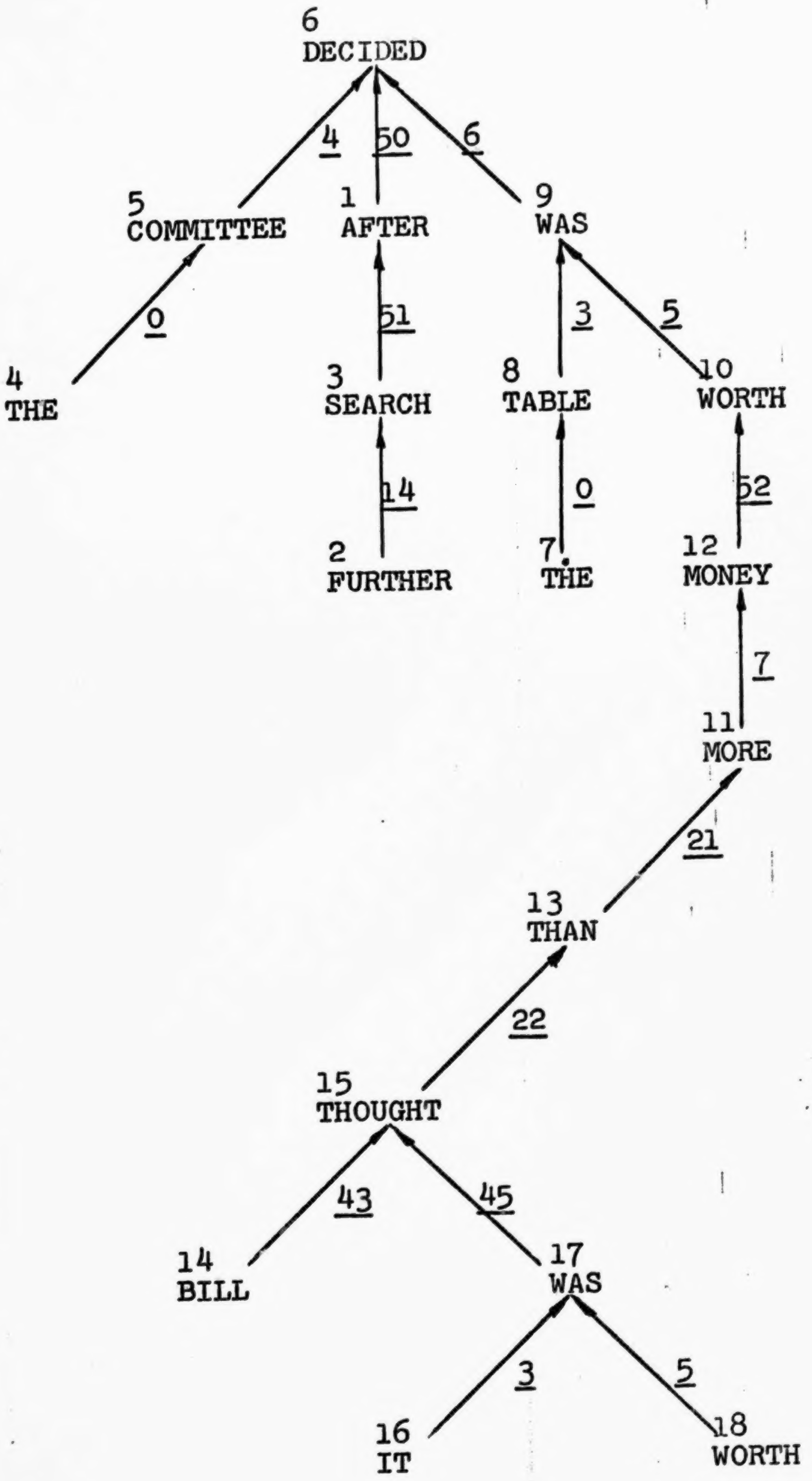
Sentence 4

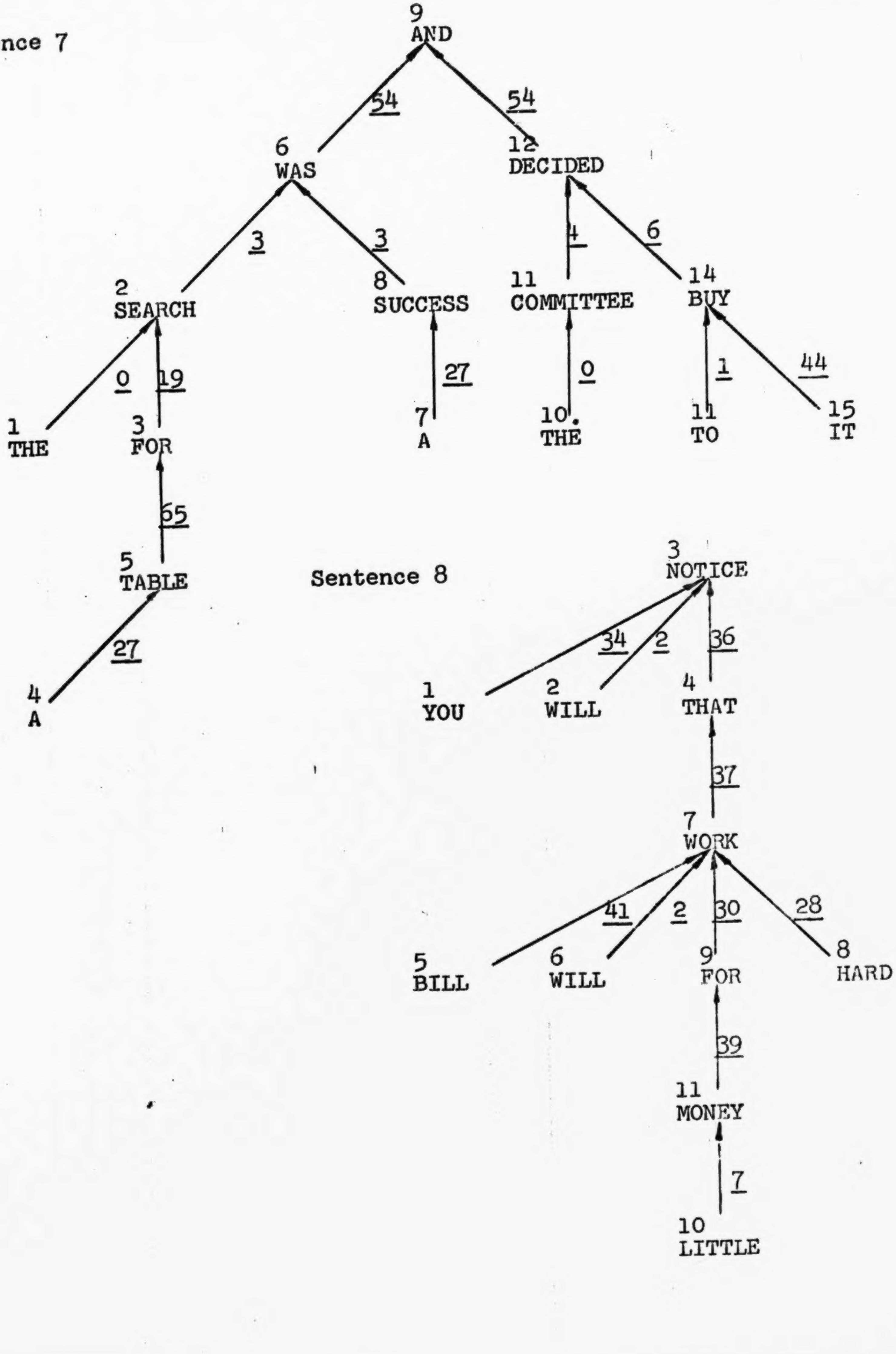

3
MAKE
33
2
62
40
1
BILL
2
WILL
7
BUY
5
TABLE
49
44
0
6
NOT
8
IT
4
THE

Sentence 5


5
DO
24
2
26
2
AND
4
WILL
8
WORK
9
9
28
30
1
BILL
3
YOU
7
HARD
9
FOR
0
32
6
THE
11
COMMITTEE
0
10
THE

Sentence 6


DECIDED
COMMITTEE
AFTER
WAS
THE
SEARCH
TABLE
WORTH
FURTHER
THE
MONEY
MORE
THAN
THOUGHT
BILL
WAS
IT
WORTH



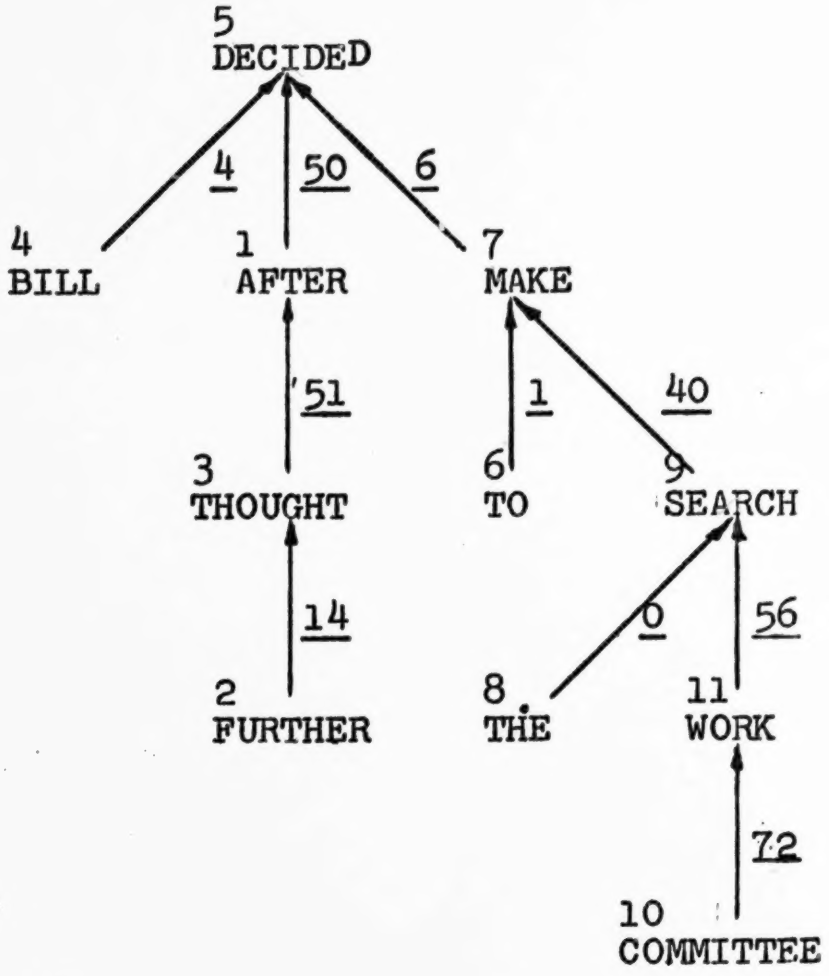
Sentence 7

9 AND
54
54
6 WAS
12 DECIDED
3
3
4
6
2 SEARCH
8 SUCCESS
11 COMMITTEE
14 BUY
0
19
27
0
1
44
1 THE
3 FOR
7 A
10 THE
11 TO
15 IT
65
5 TABLE
27
4 A

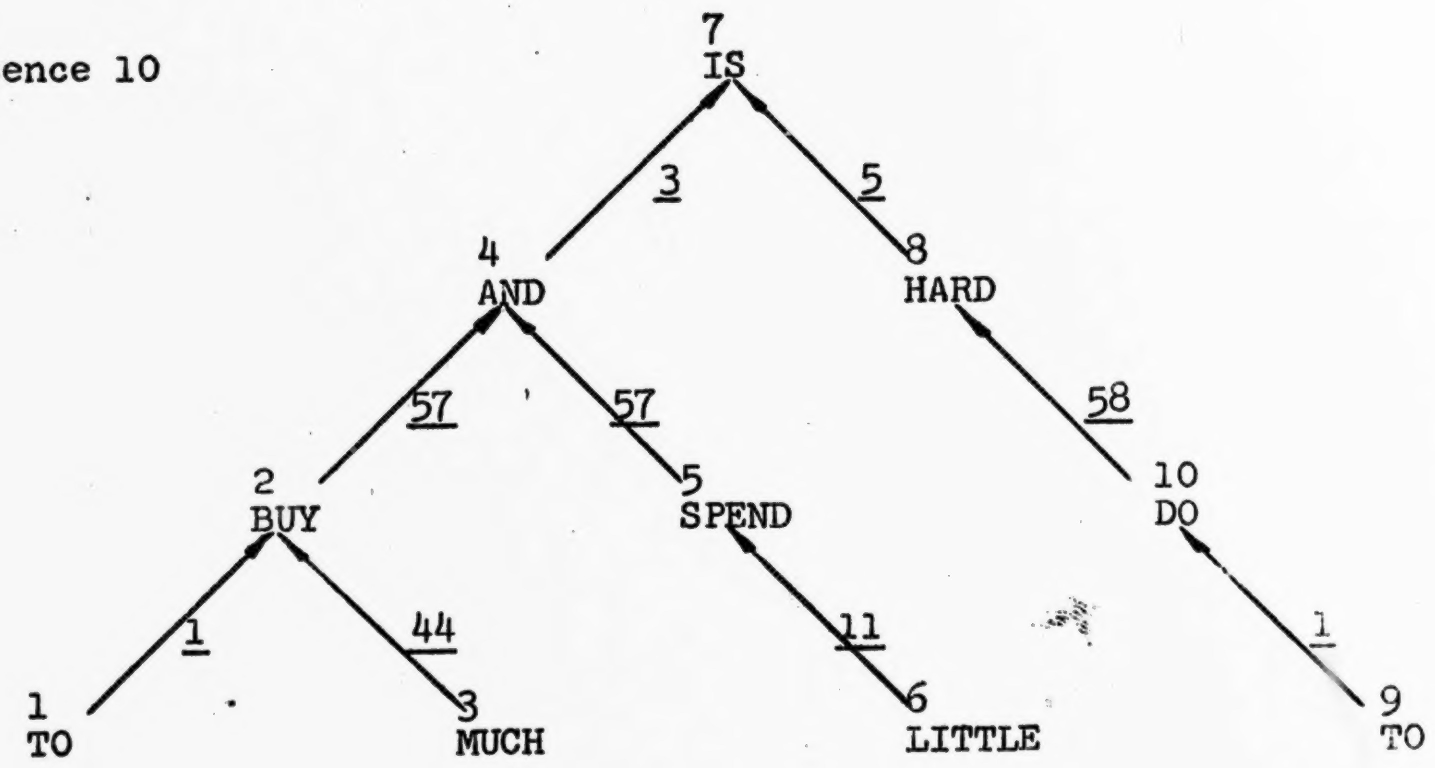
Sentence 8

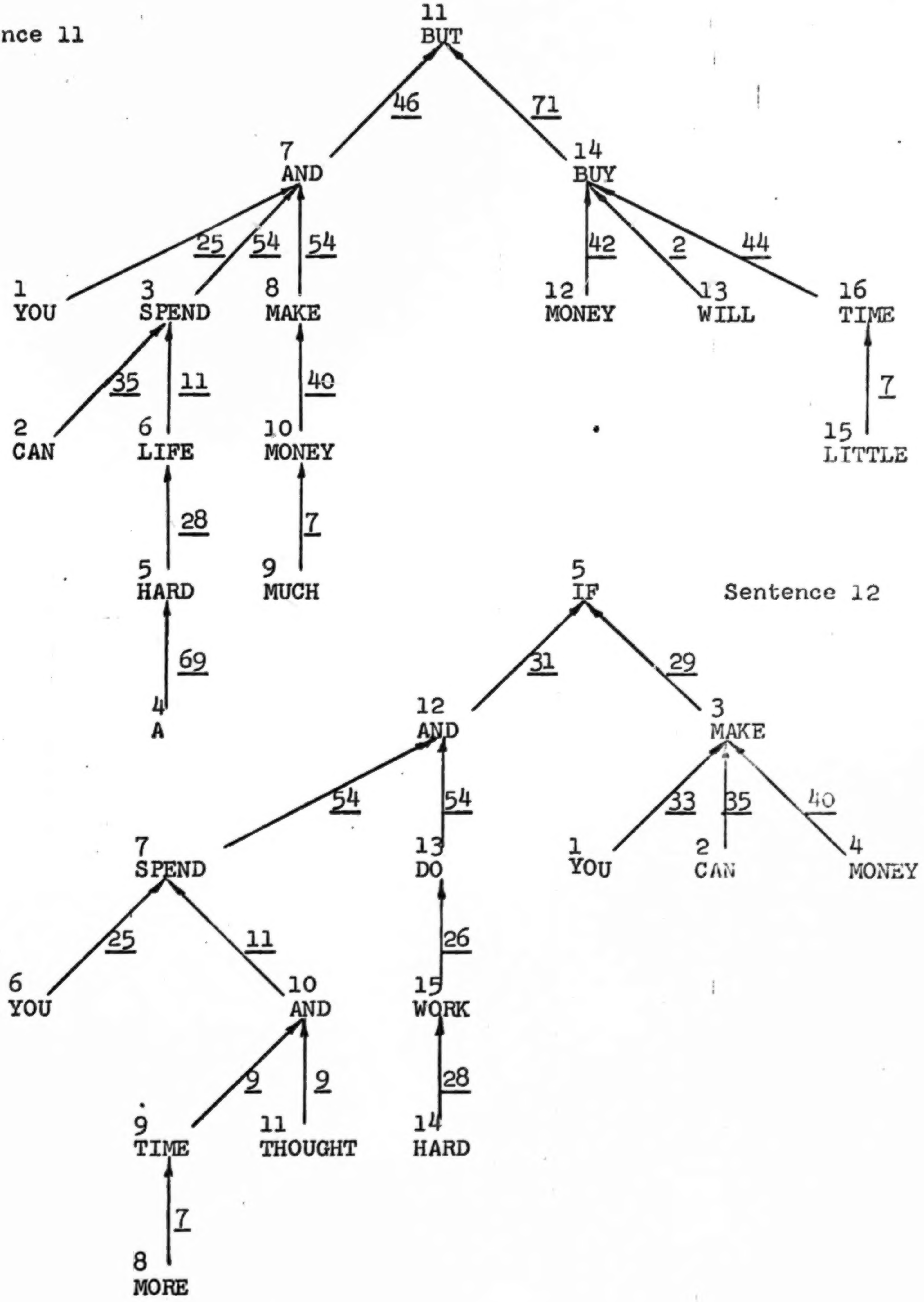
3 NOTICE
34
2
36
1 YOU
2 WILL
4 THAT
37
7 WORK
41
2
30
28
5 BILL
6 WILL
9 FOR
8 HARD
39
11 MONEY
7
10 LITTLE

Sentence 9


5
DECIDED
4
50
6
4
BILL
1
AFTER
7
MAKE
51
1
40
3
THOUGHT
6
TO
9
SEARCH
14
0
56
2
FURTHER
8
THE
11
WORK
72
10
COMMITTEE

Sentence 10


7
IS
3
5
4
AND
8
HARD
57
57
58
2
BUY
5
SPEND
10
DO
1
44
11
1
1
TO
3
MUCH
6
LITTLE
9
TO

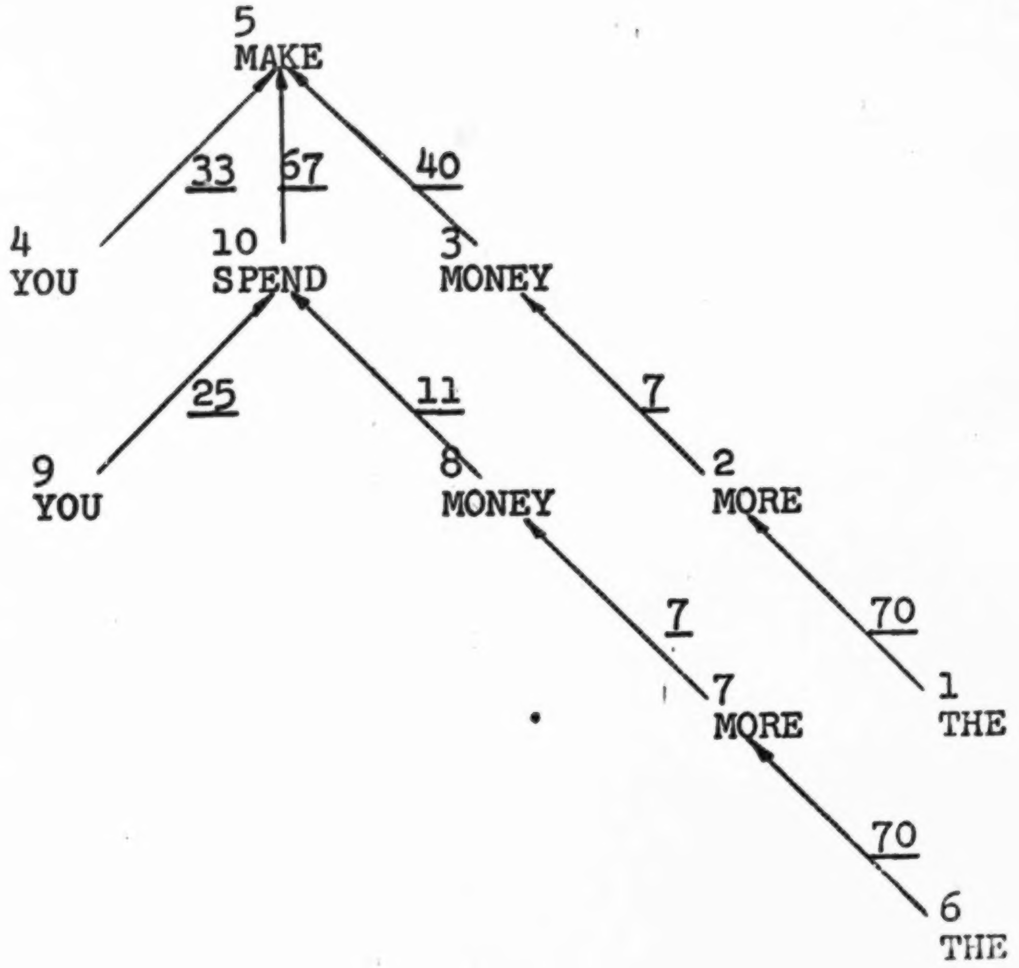


Sentence 11

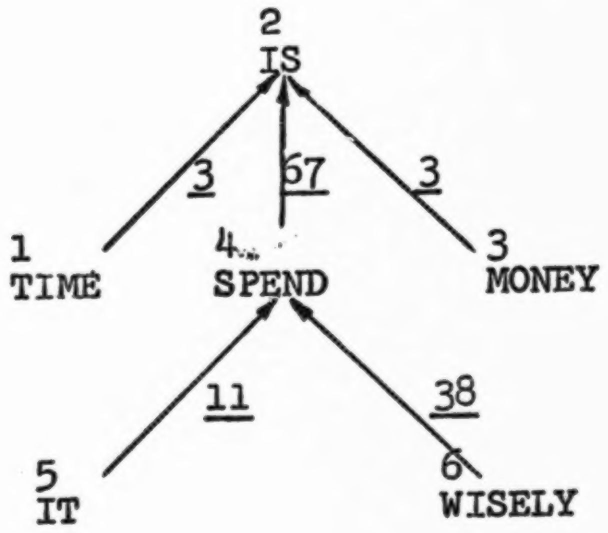
Sentence 12

11 BUT
46
71
7 AND
14 BUY
25
54
54
42
2
44
1 YOU
3 SPEND
8 MAKE
12 MONEY
13 WILL
16 TIME
35
11
40
7
2 CAN
6 LIFE
10 MONEY
15 LITTLE
28
7
5 HARD
9 MUCH
69
4 A
5 IF
31
29
12 AND
3 MAKE
54
54
33
35
40
7 SPEND
13 DO
1 YOU
2 CAN
4 MONEY
25
11
26
6 YOU
10 AND
15 WORK
9
9
28
9 TIME
11 THOUGHT
14 HARD
7
8 MORE

Sentence 13

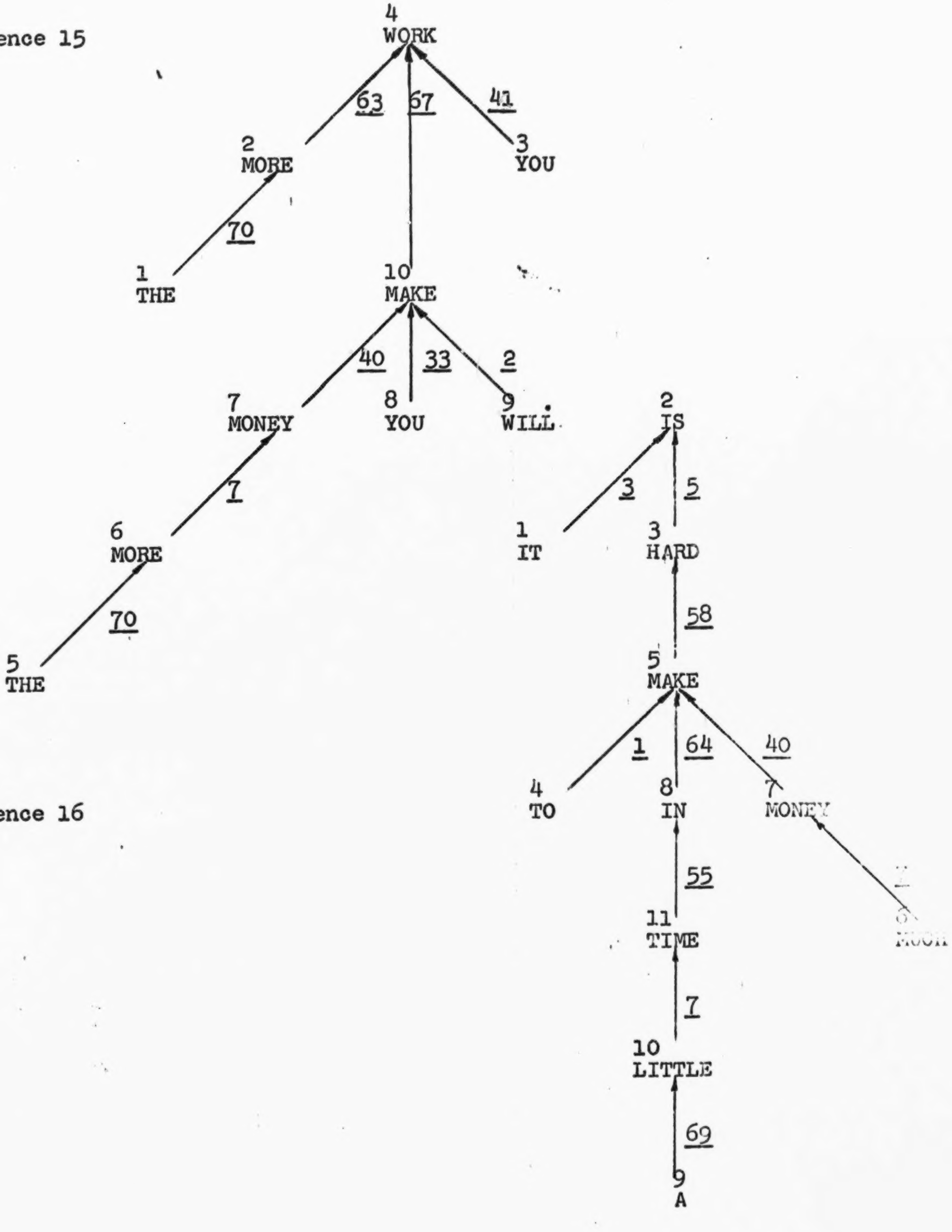

5
MAKE
33
67
40
4
YOU
10
SPEND
3
MONEY
25
11
7
9
YOU
8
MONEY
2
MORE
7
70
7
MORE
1
THE
70
6
THE

Sentence 14

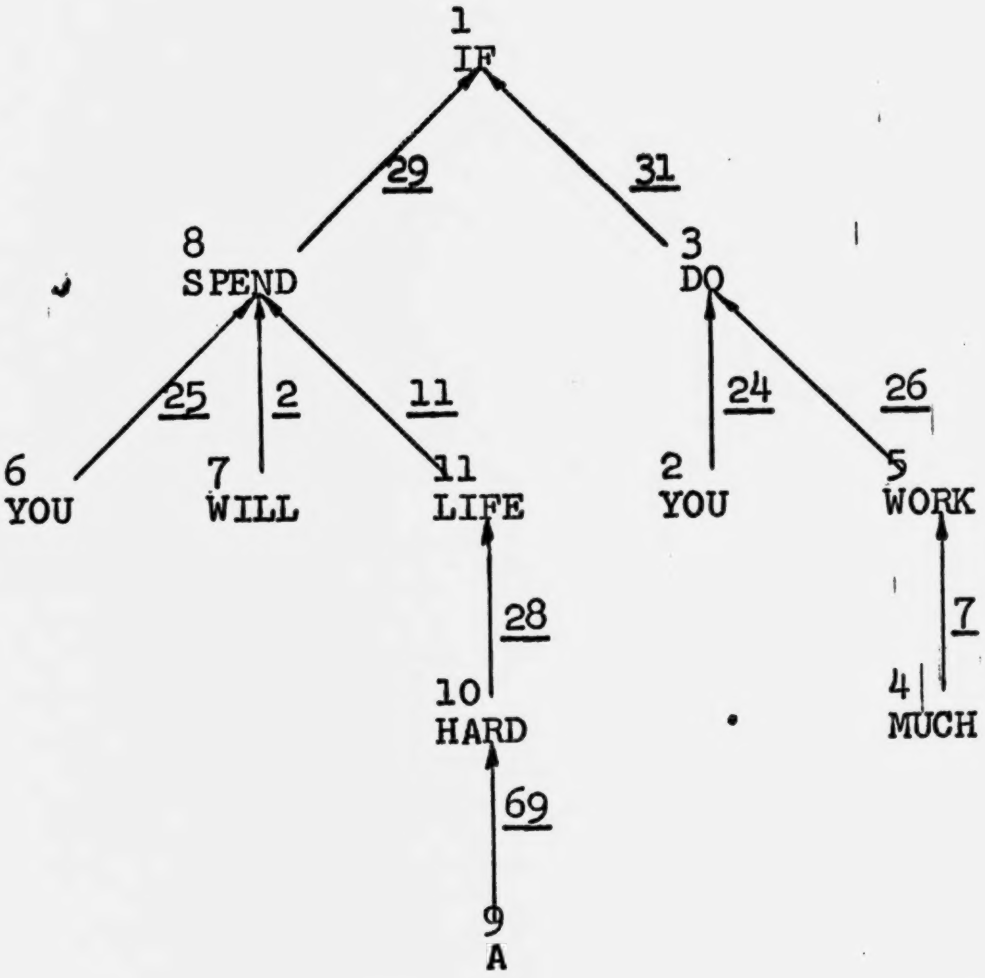

2
IS
3
67
3
1
TIME
4
SPEND
3
MONEY
11
38
5
IT
6
WISELY

Sentence 15

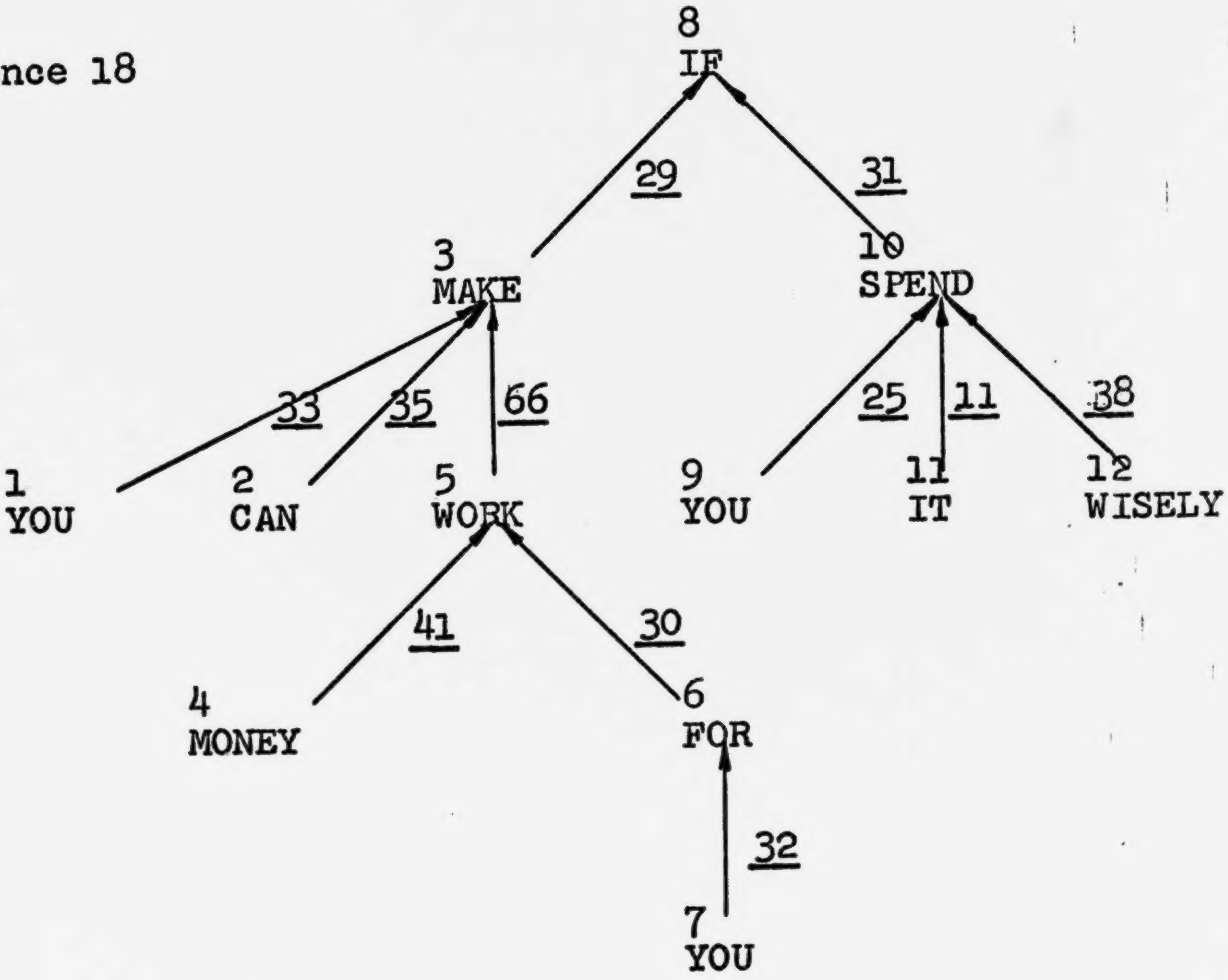
Sentence 16


4 WORK
63
67
41
2 MORE
3 YOU
70
1 THE
10 MAKE
40
33
2
7 MONEY
8 YOU
9 WILL
7
6 MORE
70
5 THE
2 IS
3
5
1 IT
3 HARD
58
5 MAKE
1
64
40
4 TO
8 IN
7 MONEY
55
11 TIME
6 MUCH
7
10 LITTLE
69
9 A

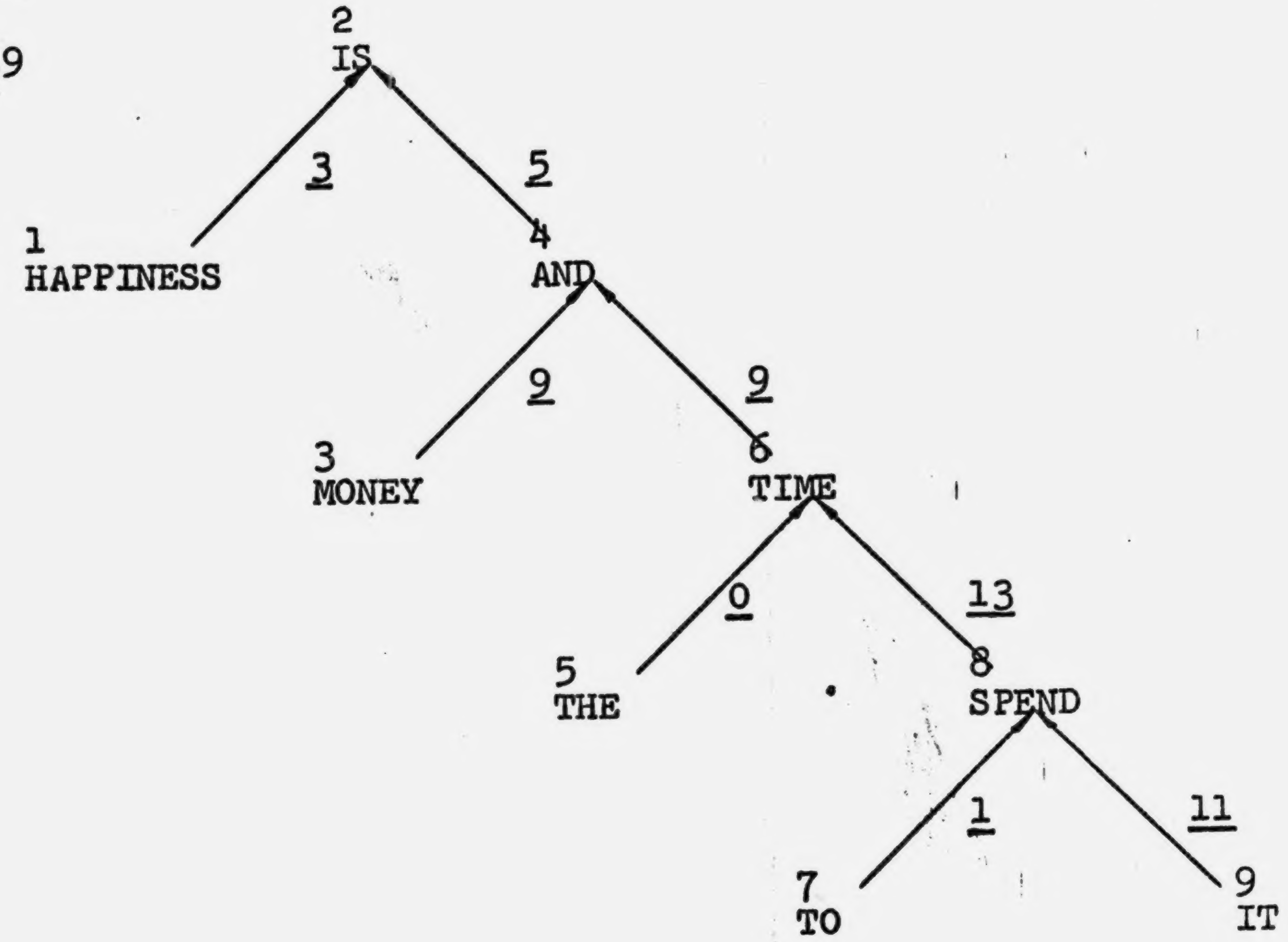
Sentence 17


1
IF
29
31
8
SPEND
3
DO
25
2
11
24
26
6
YOU
7
WILL
11
LIFE
2
YOU
5
WORK
28
7
10
HARD
4
MUCH
69
9
A

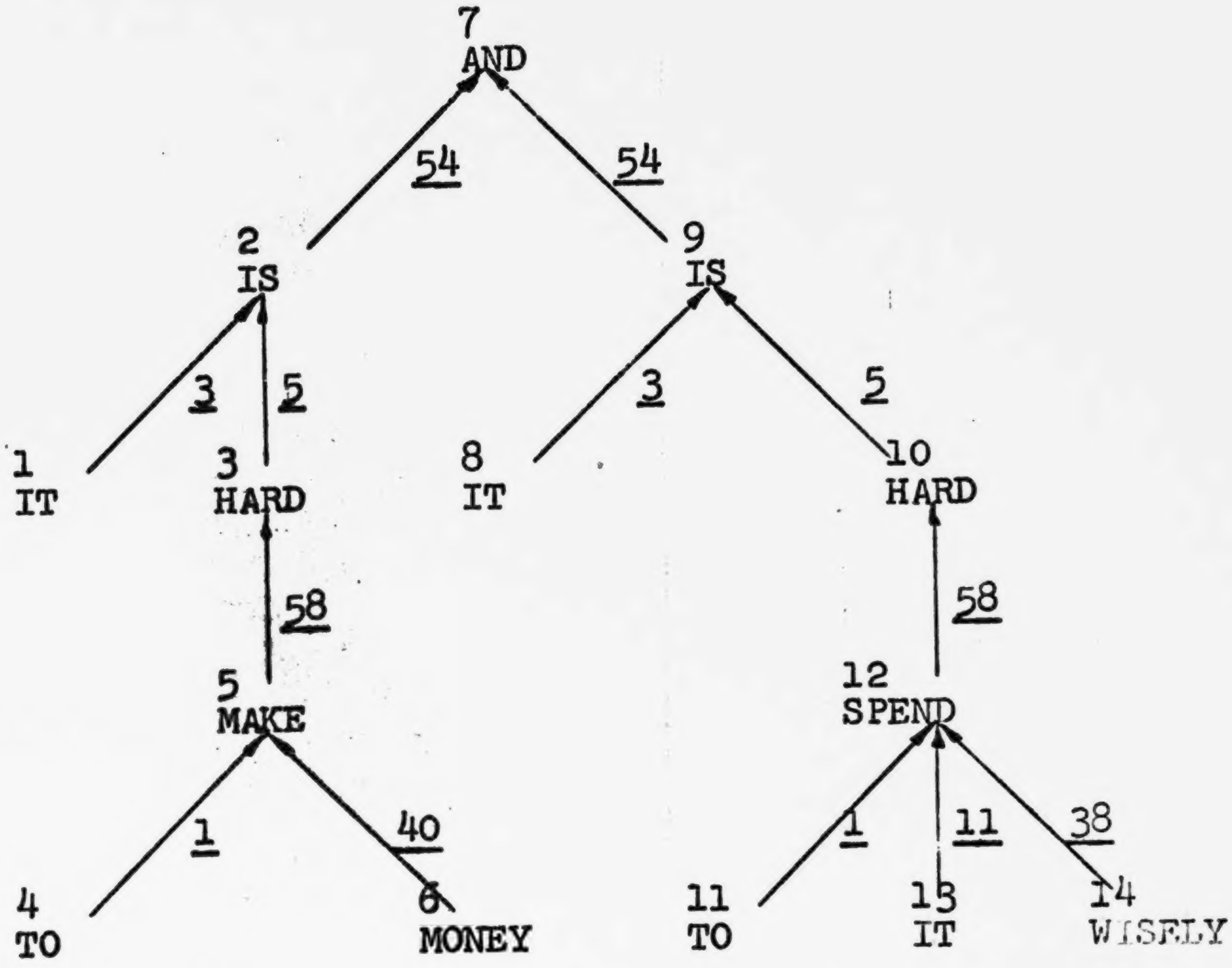
Sentence 18


8
IF
29
31
3
MAKE
10
SPEND
33
35
66
25
11
38
1
YOU
2
CAN
5
WORK
9
YOU
11
IT
12
WISELY
41
30
4
MONEY
6
FOR
32
7
YOU

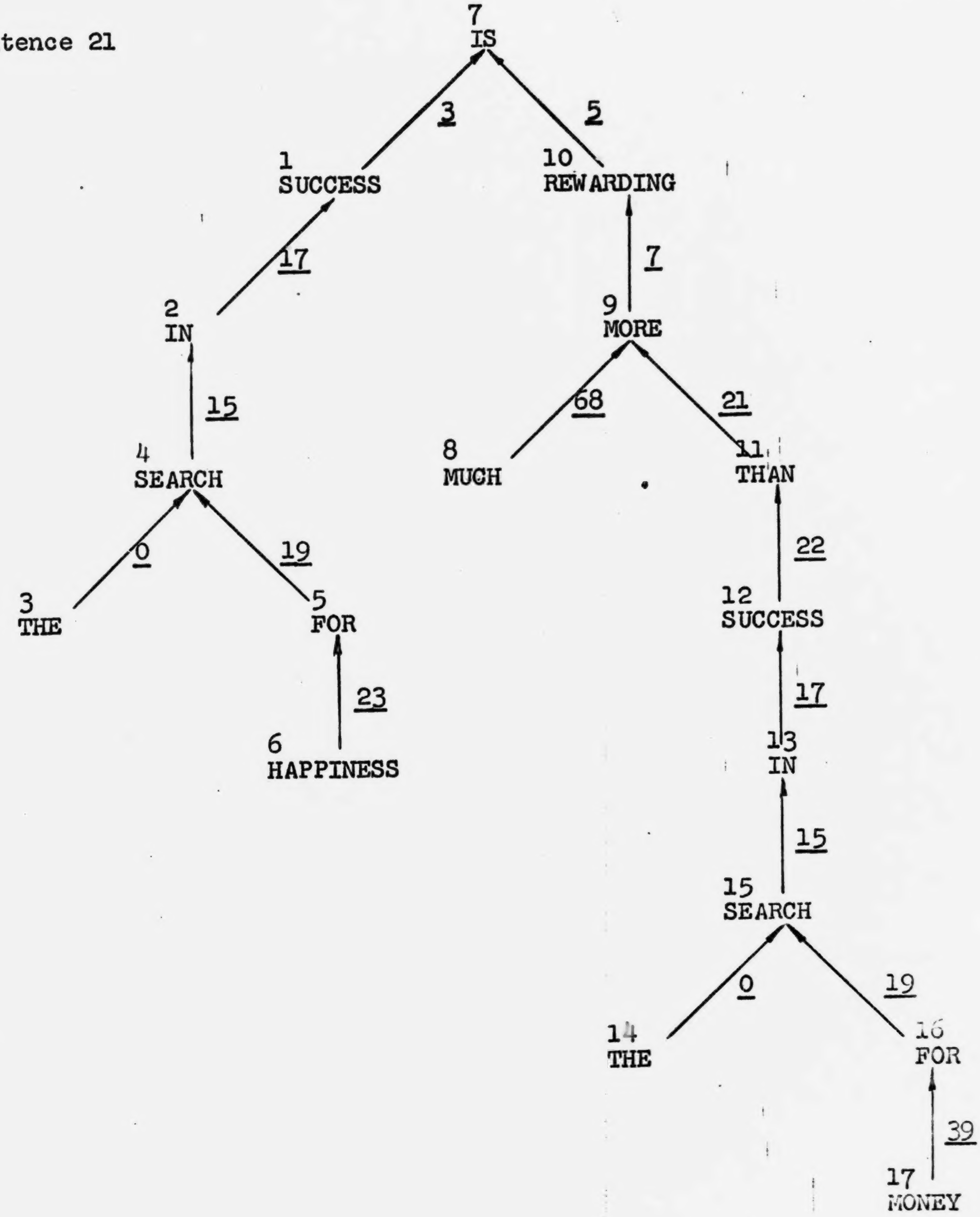
Sentence 19


2 IS
3
5
1 HAPPINESS
4 AND
9
9
3 MONEY
6 TIME
0
13
5 THE
8 SPEND
1
11
7 TO
9 IT

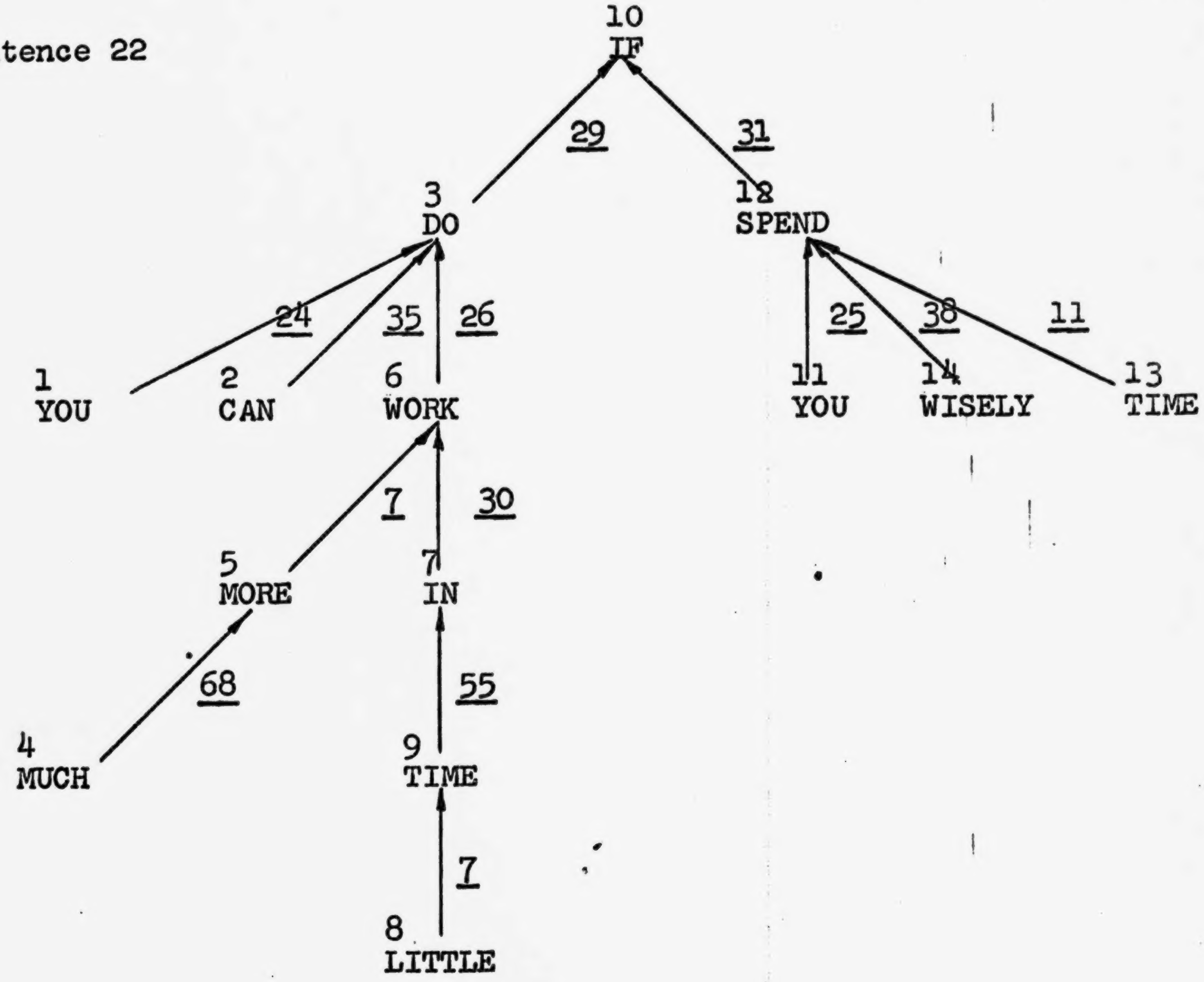
Sentence 20


7 AND
54
54
2 IS
9 IS
3
5
3
5
1 IT
3 HARD
8 IT
10 HARD
58
58
5 MAKE
12 SPEND
1
40
1
11
38
4 TO
6 MONEY
11 TO
13 IT
14 WISELY

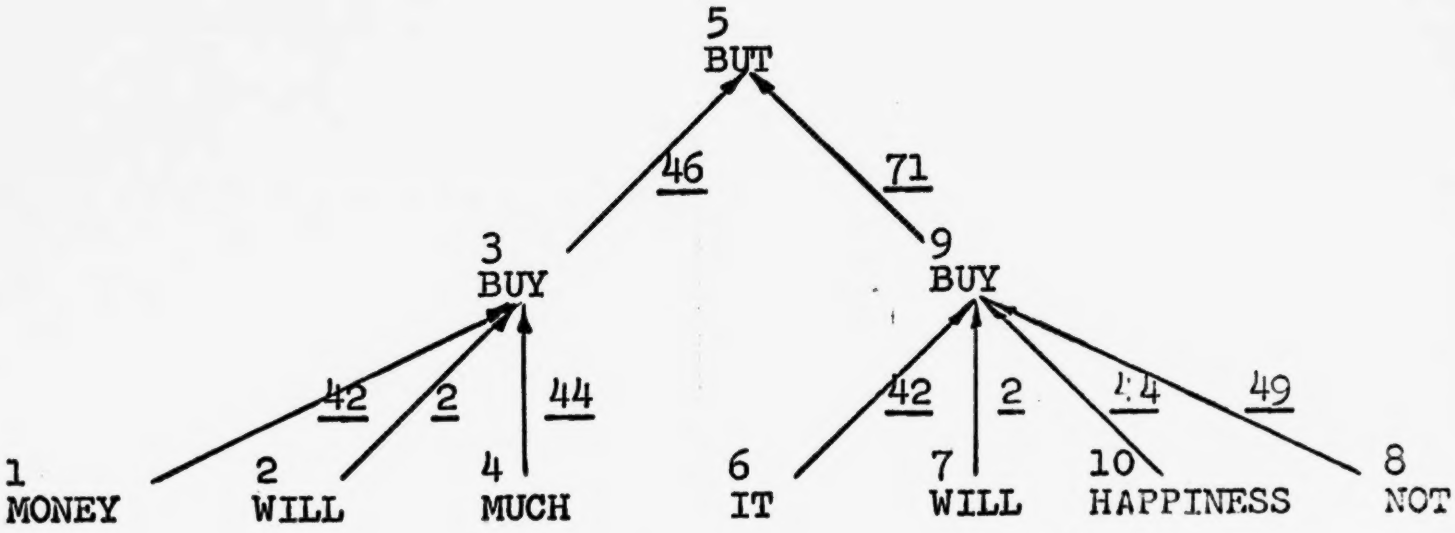
Sentence 21


IS
SUCCESS
REWARDING
IN
MORE
SEARCH
MUCH
THAN
THE
FOR
SUCCESS
HAPPINESS
IN
SEARCH
THE
FOR
MONEY

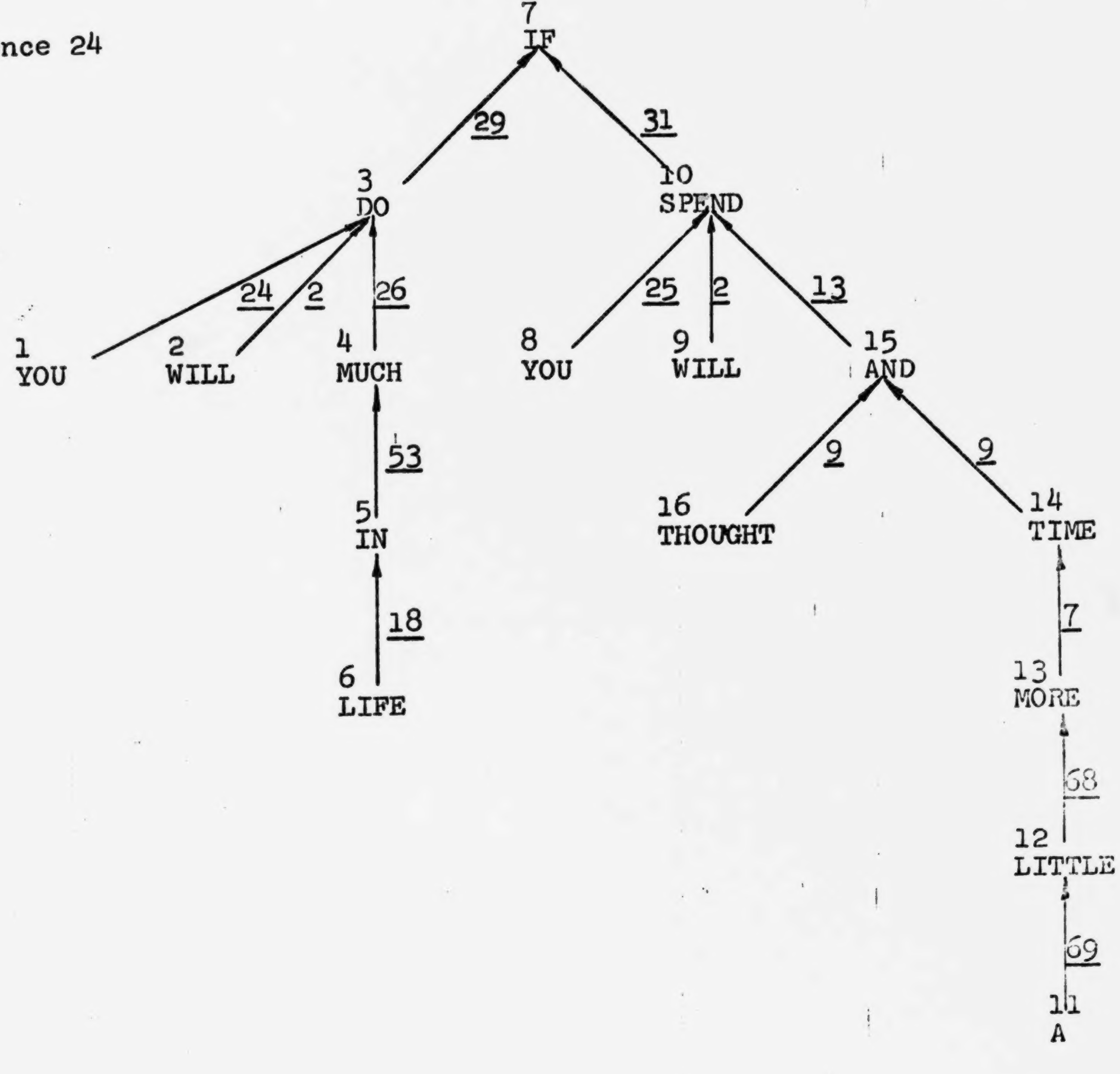
Sentence 22


10
IF
29
31
3
DO
12
SPEND
24
35
26
25
38
11
1
YOU
2
CAN
6
WORK
11
YOU
14
WISELY
13
TIME
7
30
5
MORE
7
IN
68
55
4
MUCH
9
TIME
7
8
LITTLE

Sentence 23


5
BUT
46
71
3
BUY
9
BUY
42
2
44
42
2
44
49
1
MONEY
2
WILL
4
MUCH
6
IT
7
WILL
10
HAPPINESS
8
NOT

Sentence 24


IF
DO
SPEND
YOU
WILL
MUCH
YOU
WILL
AND
IN
THOUGHT
TIME
LIFE
MORE
LITTLE
A

THE LABELS USED IN THE EXPERIMENT

Label	Sentences in which used	"Sense" of the label
F0	0,1,2,3,3,4,5,5,6,6,7,7, 9,16,19,21,21	"the"-noun. Definite article.
F1	0,2,3,3,7,9,10,10,20,20,	"to"-verb. Infinitive construction.
F2	4,5,8,8,11,15,17,23,23, 24,24	"will"-verb. Auxiliary, future.
F3	1,6,6,7,7,10,14,14,16,19, 20,20,21	noun phrase-copulative verb.
F4	0,2,3,6,7,9	noun phrase-"decided". Act of deciding.
F5	1,6,6,10,16,19,20,20,21	copulative verb-predicate adjective.
F6	0,2,3,6,7,9	"decided"-object. Denotes selection.
F7	2,2,3,3,6,8,11,11,12,13, 13,15,16,16,17,21,22,22, 24	modifier-noun or adjective. Modification delimits amount, as in "much more".
F8	0,3	"table"-"bill". Sense of setting aside for later decision.
F9	5,5,12,12,19,19,24,24	noun-"and". Conjunctive noun phrase.
F10	0	verb-"until". Temporal relation.
F11	2,3,10,11,12,13,14,17,18, 19,20,22	"spend"-noun. Sense of purchasing.
F12	0	"until"-object. Temporal relation.
F13	19,24	"spend"-noun. Sense of expending effort, time, or thought.
F14	0,6,9	noun-adjective. Sense of additional or more, as in "more time".
F15	1,21,21	"in"-noun. Sense of engaged in, as in "in search".
F16	1	verb-verb. Subordinate clause.
F17	21,21	noun-"in". As in "success in".
F18	24	"in"-noun. Sense of during, as in "in life".
F19	1,7,21,21	verb-preposition. Sense of compound verb, as in "search of".

F20	1	"on"-noun. Indication of position.
F21	6,21	"more"-"than".
F22	6,21	"than"-noun. Sense of comparing.
F23	2,21	"for"-noun. Sense of a quest, as in "look for".
F24	5,17,22,24	subject-verb ("do"). Subject may be compound, headed by conjunction.
F25	11,12,13,17,18,22,24	pronoun-verb. Sense of actor-action.
F26	5,12,17,22,24	"do"-noun. Sense of action-object.
F27	7,7	"a"("an")-noun. Indefinite article.
F28	5,8,11,12,17	adjective-noun. Sense of difficult, as in "hard life".
F29	12,18,22,24,17	"if"-Y in context "if X then Y".
F30	5,8,18,22	noun-preposition. As in "work for".
F31	12,18,22,24,17	"if"-X in context "if X then Y".
F32	5,18	"for"-noun. Sense of in behalf of.
F33	4,12,13,15,18	subject-"make".
F34	8	subject-"notice". Sense of taking note of.
F35	11,12,18,22	"can"-verb. Auxiliary, indicating possibility.
F36	8	verb-"that". Relative pronoun.
F37	8	"that"-work. Relative pronoun.
F38	14,18,20,22	verb-adverb. As in "spend wisely".
F39	8,21	"for"-noun. Sense of to obtain, as in "for money".
F40	4,9,11,12,13,15,16,20	"make"-object.
F41	8,15,18	subject-"work". Sense of doing.
F42	11,23,23	subject-"buy".
F43	6	subject-"thought". Sense of being under the impression that.
F44	4,7,10,11,23,23	"buy"-object.

F45	2,6	verb-verb. Connecting clauses with an understood relative pronoun.
F46	11,23	X-"but"in context "you can X, but".
F47	2	adjective-noun. As in "rewarding work".
F48	2	"on"-noun. Sense of engaged in.
F49	3,4,23	"not"-verb. Auxiliary, negation.
F50	6	verb-"after". Sense of following in time.
F51	6,9	"after"-noun. Sense of following in time.
F52	6	"worth"-object. Sense of valuation.
F53	24	"much"-"in".
F54	3,3,7,7,11,11,12,12,20,20	word-"and". Conjunction.
F55	16,22	"in"-"time".
F56	9	"search"-object. Action-object.
F57	10,10	verbal noun-"and".
F58	10,16,20,20	"hard"-verb. Sense of difficult.
F59	3	verb-"for". Temporal sense.
F60	3	"for"-noun. Temporal sense.
F61	2	noun-"on". Sense of assigning to, as in "time on".
F62	4	verb-verb. Disjunction of clauses.
F63	15	verb-"more".
F64	16	verb-"in". Temporal relation.
F65	7	"for"-noun. As in "search for X".
F66	18	verb-verb. As in (you can)make (money) work.
F67	13,14,15	verb-verb.
F68	3,21,22,24	adverb-adjective. Sense of delimiting amount, as in "little more".
F69	3,11,16,17,24	"a"-adverb. As in "a little".

F70	5,13,13,15	"the"-nominal adjective. As in "the more the merrier".
F71	11,23	"but"-Y in context "You can X,but Y".
F72	9	"committee"-"work".

SYNTAX/SEMANTICS TABLE FOR CORPUS USED IN THE EXPERIMENT

Word	Symbol	Labels word may be value of (and precede if *)	Labels word may be symbol for (and precede if *)
A	A11	*27,*69	
AFTER	A43	*50	*51
AND	A3	*3,5,6,11,13,*24,31,*46	*9,*54,*57
BILL	A36	*4,8,*9,*33,*41,*43	0
BUT	A15		46,*71
BUY	A1	6,*46,*57,62,71	1,2,42,*44,49
CAN	A10	*35	
COMMITTEE	A32	*4,32,*72	0
DECIDED	A33	45,54	4,*6,50
DO	A8	*29,31,54,58	1,2,24,*26,35
FOR	A26	19,30,59	*23,*32,*39,*60,*65
FURTHER	A38	*14	
HAPPINESS	A27	*3,23,44	
HARD	A7	5,*28	0,*58,69
IF	A19		*29,*31
IN	A18	5,17,30,53,64	*15,*18,*55
IS	A6	*54	*3,*5,*67
IT	A24	*3,11,*42,44	
LESS	A42	*7	
LIFE	A12	11,18	28
LITTLE	A5	*7,11,*68	69
MAKE	A13	6,*29,54,58,67	1,2,33,35,*40,*62,*64,*66,*67
MONEY	A14	3,*9,11,39,*40,*41,*42,52	7
MORE	A21	*7,*63	*21,68,70

SYNTAX/SEMANTICS TABLE FOR CORPUS USED IN THE EXPERIMENT

Word	Symbol	Labels word may be value of (and precede if *)	Labels word may be symbol for (and precede if *)
MUCH	A2	*7,26,44,*68	*53
NOT	A34	*49	
NOTICE	A39	*3,12	0,2,14,*16,34,*36
OF	A46	19	
ON	A41	5,61	*20,*48
REWARDING	A30	5,*47	7
SEARCH	A29	*3,15,40,51	0,14,*19,*56
SPEND	A4	6,13,31,*54,57,58,67,29	1,2,*11,*13,25,35,*38,49
SUCCESS	A28	*3,22	*17,27
TABLE	A35	*3,6,20,40,54,65	0;1,*8,*10,27,*59
THAN	A31	21	*22
THAT	A47	36	*37
THE	A23	*0,*70	
THOUGHT	A20	9,22,51	14,43,*45
TIME	A17	*3,*9,11,44,55,60	0,7,*13,*61
TO	A48	*1	
UNTIL	A37	10	*12
WAS	A45	6,45,*54	*3,*5
WERE	A40	16	3,*5
WILL	A16	*2	
WISELY	A25	38	
WORK	A22	26,37,48,56,66	2,7,*28,*30,41,47,63,*67,72
WORTH	A44	5	*52
YOU	A9	*3,9,*24,*25,32,*33,*34,*41	

REFERENCES

Backus, J. W. The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference. Information processing. Paris: UNESCO, 1959, 125-132.

Bobrow, D. Syntactic analysis of English by computer--a survey. Proc. 1963 Fall Joint Computer Conference. Baltimore: Spartan Books, 1963. 365-387.

Chomsky, N. Syntactic structures. The Hague, The Netherlands: Vitgverij Mouton, 1957.

Chomsky, N. On certain formal properties of grammars. Information and control, 1959, 2, 137-167.

Chomsky, N. On the notion "rule of grammar". In R. Jakobson (Ed.) Structure of language and its mathematical aspects. Proc. of symposia in applied mathematics. Providence: American Mathematical Society, 1961, Pp 6-24.

Duncker, K. On problem solving. Psychol. Monogr., 1945, 58, whole no. 270 (translated by L. S. Lees from 1935 original).

Feigenbaum, E. A. An information processing theory of verbal learning. The RAND Corporation, Mathematics Division, P-1817, 1959.

Hays, D. G. & Ziehe, T. Studies in machine translation--10: Russian sentence structure determination. The RAND Corporation, RM--2538, 1960.

Hays, D. G. Basic principles and technical variations in sentence-structure determination. In Cherry, C. (Ed.) Proceedings of the Fourth London Symposium on Information Theory. London: Butterworths, 1961. Pp 367-376.

Hockett, C. F. Grammar for the hearer. In Jakobson, R. (Ed.) Structure of language and its mathematical aspects. Proceedings of symposia in applied mathematics, Vol. 12. Providence: American Mathematical Society, 1961, Pp 220-236.

Irons, E. T. A syntax-directed compiler for ALGOL-60. Comm. ACM, 1961, 4, 51-55.

Knowlton, K. C. Sentence parsing with a self-organizing heuristic program. Doctoral Dissertation, Mechanical translation group, Research Laboratory of Electronics, MIT, 1962.

Lindsay, R. K. Toward the development of a machine which comprehends. Doctoral dissertation, Carnegie Institute of Technology, 1961.

Lindsay, R. K. Inferential memory as the basis of machines which understand natural language. In Feigenbaum, E. A. and Feldman, J. (Eds.) Computers and thought. New York: McGraw-Hill, 1964, Pp. 217-233.

Lindsay, R. K. JIGSAW-1. Information processing report No. 13, The University of Texas, 1964.

Lindsay, R. K., Dauwalder, J. H., Pratt, T., & Shavor, K. An associative memory organizing system. (In preparation).

Newell, A. (Ed.) *Information processing language V manual*, (2nd ed.) Englewood-Cliffs: Prentice Hall, 1964.

Oettinger, A. G. Automatic syntactic analysis and the pushdown store. In Jakobson, R. (Ed.) *Structure of language and its mathematical aspects*. Proceedings of symposia in applied mathematics. Providence: American Mathematical Society, 1961, *12*, Pp. 104-129.

Oettinger, A. G. & Kuno, S. Syntactic structure and ambiguity of English. *Proc. 1963 Fall Joint Computer Conference*. Baltimore: Spartan Books, 1963, 397-418.

Pendergraft, E. Report No. 16 Final report on machine language translation study. Linguistics Research Center. LRC63-P16. June 1963. The University of Texas.

Raphael, B. SIR: a computer program for semantic information retrieval. Doctoral dissertation, MIT, 1964.

Rhoades, I. A new approach to the mechanical translation of Russian. *National Bureau of Standards Report No. 6295*, 1959.

Simon, H. A. Extensions of the heuristic compiler. CIP Working Paper #45, Carnegie Institute of Technology, 1962.

Sommers, F. The theory of types for natural languages. 1961 (mimeograph).

Thompson, F. The semantic interface in man-machine communication. TEMPO, The General Electric Company RM 63 TMP-35, 1963.

Yngve, V. H. A model and an hypothesis for the language production. *Proc. Amer. Philos. Soc.*, 1961, *104*, 444-466.